(19) Europäisches Patentamt
European Patent Office
Office européen des brevets

(11) Publication number: **0 199 173**
**A2**

# (12) EUROPEAN PATENT APPLICATION

(21) Application number: 86104747.0

(22) Date of filing: 08.04.86

(51) Int. Cl.⁴: G06F 9/44 , G06F 9/42 , G06F 9/46

(30) Priority: 08.04.85 JP 72646/85
14.08.85 JP 177542/85

(43) Date of publication of application:
29.10.86 Bulletin 86/44

(84) Designated Contracting States:
DE FR GB IT

(71) Applicant: HITACHI, LTD.
6, Kanda Surugadai 4-chome Chiyoda-ku
Tokyo 100(JP)

(72) Inventor: Kida, Hiroyuki
17-1-102, Moriyamacho-3-chome
Hitachi-shi(JP)
Inventor: Maejima, Hideo
26-2, Nakanarusawacho-2-chome
Hitachi-shi(JP)
Inventor: Masuda, Ikuro
15-7, Moriyamacho-2-chome
Hitachi-shi(JP)
Inventor: Baba, Shirou
39-2-407, 5043, Yamaguchi
Tokorozawa-shi(JP)

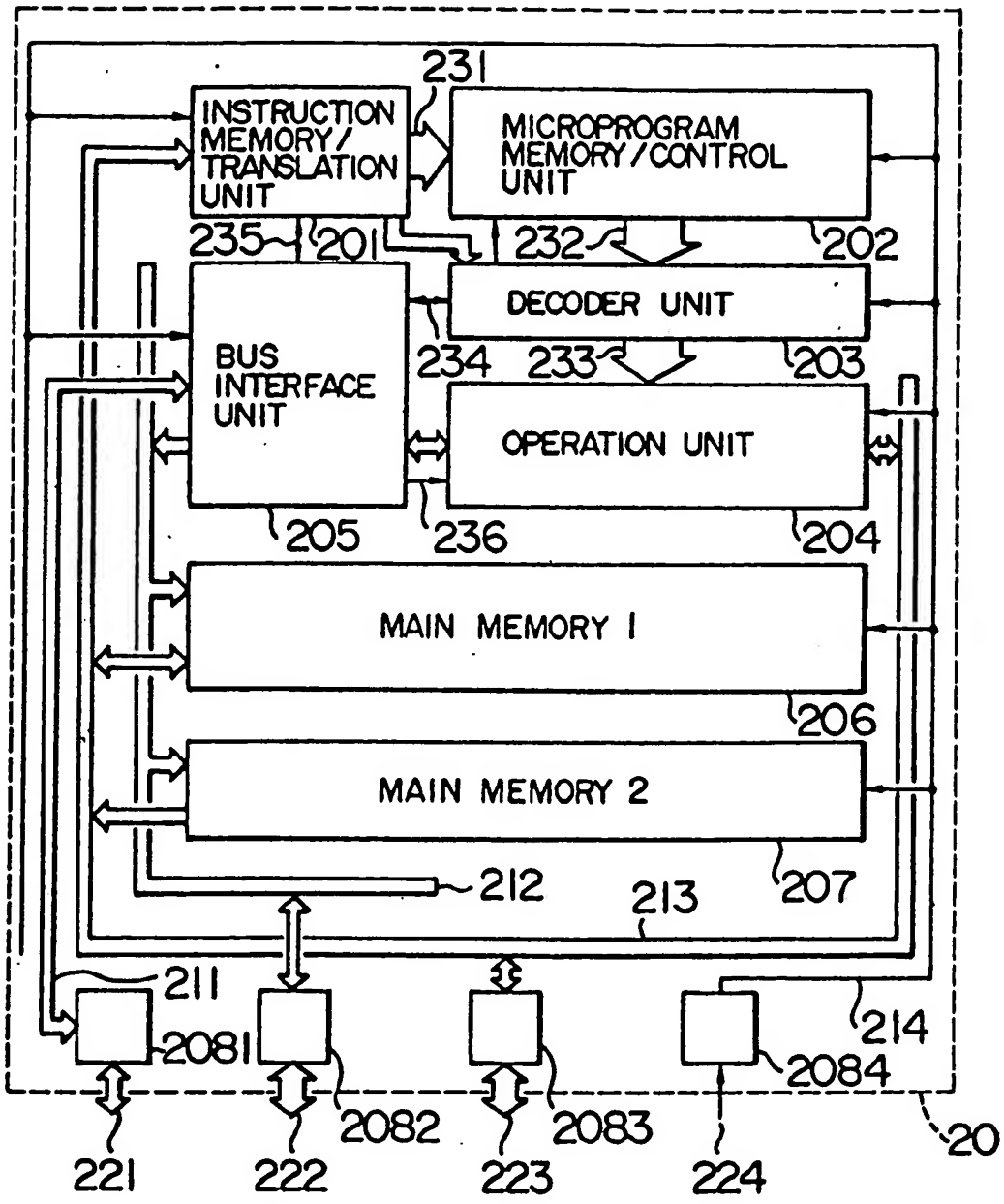
(74) Representative: Patentanwälte Beetz sen. - Beetz jun. Timpe - Siegfried - Schmitt-Fumian
Steinsdorfstrasse 10
D-8000 München 22(DE)

## (54) Data processing system.

(57) A data processing system incorporating a main memory (206) for storing instructions and operands and performing data processing in a mode of microprogram control system in response to instructions read out of the main memory (206). The system translates an instruction word read out of the main memory (1, 2) into an intermediate machine word having the orthogonal format, and addresses a microprogram memory (202) in correspondence to the instruction word by analyzing the intermediate machine word. The system further incorporates a plurality of register sets so that each different task can use an individual register set, and a memory (2106) for memorizing the number of registers holding parameters used commonly among procedures corresponding to the register sets, so that the number of registers for each use can be changed arbitrarily for each register set by using the memory.

EP 0 199 173 A2

FIG. 1


231
INSTRUCTION MEMORY/ TRANSLATION UNIT
MICROPROGRAM MEMORY/CONTROL UNIT
235
201
232
202
DECODER UNIT
BUS INTERFACE UNIT
234
233
203
OPERATION UNIT
205
236
204
MAIN MEMORY 1
206
MAIN MEMORY 2
212
213
207
211
2081
214
2084
2082
2083
20
221
222
223
224

# DATA PROCESSING SYSTEM

## BACKGROUND OF THE INVENTION

### FIELD OF THE INVENTION

The present invention relates to a data processing system and, particularly, to a microprogram-controlled data processing system incorporating an instruction decoder and multiple register sets, the system being suitable for constructing a microcomputer.

### DESCRIPTION OF THE RELATED ART

Recent advanced micro-miniaturizing technology in the field of semiconductor devices, particularly MOS (Metal Oxide Semiconductor) devices has enabled upgraded functions and performance of microcomputers. In order to avoid the complexity of logics resulting from the enhanced circuit integration, the achievement of an integrated circuit by an orderly structured logic circuit is becoming the mainstream method. One practice is the microprogram control system. A microprogram-controlled data processing system is generally constructed using an instruction decoder which generates the microprogram ROM address from the instruction word. However, this system necessitates a large scale hardware for the instruction decoder, and a microprogram control system coping with this problem by elimination of the instruction decoder is proposed in Japanese Patent Unexamined Publication No. 57-2030141. However, this system sets the instruction word directly in part of the microprogram ROM address information, and therefore as the instruction word length increases from 8 bits to 16 bits and to 32 bits the length of the microprogram ROM address increases with it, resulting disadvantageously in an expanded microprogram ROM address decoder.

Another hardware scheme for upgrading the performance of microcomputers is the general-purpose register system, in which a microcomputer is provided therein with many registers so that various operations take place among registers with the intention of high-speed processing. However, if the program includes frequent task switching such as procedure call/return, the contents of the general-purpose registers need to be saved and resumed to/from the stack (a first-in-last-out memory) frequently at each switching. The time used for the saving and resuming operations increases the total processing time, and speedup of processing can be hampered. The time used for the saving and resuming operations is enormous for a system having a large number of general-purpose registers.

As a means for overcoming the problem, there has been known the multiple register sets system, in which a plurality of register sets are provided and a register set is used for each task by switching, as described in IEEE MICRO, Vol. 2, No. 4, p. 13, Nov. 1982. This system allows the avoidance of operation for saving the register contents at each procedure call and for restoring the original parameter at each procedure return. Moreover, this system does not necessitate the parameter transaction among procedures, and as a result high-speed register saving/resuming processing can be accomplished. On the other hand, however, only a small part of many register sets is used by application programs with a procedure nesting in relatively small depth, with the result of inefficient use of the hardware resources. In addition, since the number of registers used for parameter transaction among procedures and the number of registers without connection among procedures are each fixed, the processing, for example, of a procedure requiring an extremely large number of the latter-type registers will be forced to use a memory area in substitution for the lacking register even though many of the former-type registers are left unused. Accordingly, this system also has the problem of insufficient use of the hardware resources.

## SUMMARY OF THE INVENTION

An object of this invention is to provide a data processing system having enhanced functions and performance.

Another object of this invention is to provide a data processing system of the type of instruction decoding system which is responsive to various instruction formats.

A further object of this invention is to provide a versatile and flexible data processing system of the type of multiple register sets system capable of defining arbitrarily the number of registers for storing parameters used commonly among procedures and the number of registers for storing data specific to each procedure.

The inventive data processing system is of the type of instruction decoding system in which an instruction word (machine word) read out of the main memory is arranged into an intermediate machine instruction word in the orthogonal format suitable for the decoding hardware arrangement and

then the intermediate machine word is decoded. An instruction word in the orthogonal format is defined here to be an instruction word having information bits representing operand specifiers for identifying an operation code spesifying the operation to be executed and locations in the main storage containing operands. In a data processing system in general, the bit positions for the operation code and operand specifying differ depending on each instruction so that information on the operation and operands is contained in the limited instruction word with less redundancy. In the conventional system, this type of instruction has been decoded or used for microprogram ROM address generation, and therefore the system needs to incorporate a large scale address generating hardware for the decoders and the like.

The inventive system performs instruction decoding by initially converting an instruction word into an instruction word of the orthogonal format suitable for decoding, noting the facts that the bit arrangement of instruction words which are not orthogonal is not completely random, but has some rule, and translation of instruction words not orthogonal into orthogonal instruction words is relatively easy, and the decoding hardware and ROM capacity can be reduced when the microprogram ROM address is generated by decoding an orthogonal instruction word.

The inventive system incorporates decoders for arranging instruction words into intermediate machine words separately for each instruction format so that decoding of intermediate machine words takes place commonly for every instruction format, with the intention that instruction words of different formats or different word length can be processed.

Having the foregoing arrangement, the inventive system can deal with instruction words in different formats without modifying or adding the hardware arrangement within the processor, except for the instruction word storage and the translation unit which arranges instruction words into intermediate machine words for decoding, and the hardware arrangement used for instruction decoding can be reduced to 1/5 to 1/10 of the counterpart of the conventional instruction decoder system.

As another feature of this invention, the system can process instruction words having at least two instruction formats.

With the intention of efficient use of the general-purpose registers, the inventive system is provided with instruction-controlled registers and memory capable of arranging a register set in correspondence to the contents of the registers, so that register sets for procedures are placed in the memory in accordance with the register contents. The number of register sets provided in the memory is made variable depending on each process, whereby the registers are used efficiently. The remaining portion of the memory where register sets are not placed can be used as part of the main memory, whereby the hardware resources can be used efficiently.

A single register in a physical sense is used to set data used commonly by several tasks in transacting parameters among procedures so as to eliminate the need of data transfer among registers when the procedure is changed, whereby overhead of data processing is reduced and the processing speed of the data processing system is enhanced.

The inventive system merely needs the switching of register sets at the switching of tasks caused by the occurrence of interrupt events, eliminating the need for data saving and restoration to/from the main memory or the like, whereby tasks can be switched quickly and the processing speed can be enhanced.

The ability of arbitrary setting common access areas for tasks by the instruction eliminates the idle time spent for parameter transaction, whereby the processing speed can be enhanced. Designation of a register local area in arbitrary size for each procedure increases the latitude of choice of registers, whereby software can be simplified. The number of tasks can be set by the instruction, and a system best fit for the user's application can be constructed. Register areas left unused by tasks can be used as a main memory area, which creats a function resembling a single-chip microcomputer with an on-chip RAM area and provides the ability to construct a user system flexibly.

As described above, the present invention improves the decoding function for instructions in reading the microprogram RAM without increasing the scale of hardware. The invention can provide a high performance, high functioning data processing system without a significant increase in the hardware arrangement, but through the efficient use of the hardware resources while employing the general-purpose register system.

## BRIEF DESCRIPTION OF THE DRAWINGS

These and other objects and advantages of the present invention will become apparent by making reference to the following description and accompanying drawings, in which:

Fig. 1 is a block diagram showing the processor in the data processing system embodying the present invention;

Fig. 2 is a block diagram showing the microcomputer formed on a single semiconductor substrate by application of the invention data processing system;

Fig. 3 is a block diagram showing the arrangement of the instruction word memory/translation unit, microprogram memory control unit, decoder unit, and operation unit included in the processor shown in Fig. 1;

Fig. 4 is a set of diagrams showing examples of the instruction format;

Fig. 5 is a block diagram showing another embodiment of the instruction word memory/translation unit in the processor shown in Fig 1;

Fig. 6 is a block diagram showing still another embodiment of the instruction word memory/translation unit in the processor shown in Fig. 1;

Fig. 7 is a block diagram showing an embodiment of the instruction word memory/translation unit derived from Fig. 6;

Fig. 8 is a block diagram showing separately the functions of the instruction word memory/translation unit in Fig. 3;

Fig. 9 is a block diagram showing an embodiment of the instruction word memory/translation unit shown in Fig. 8;

Fig. 10 is a timing chart showing the signals at various portions of the instruction word memory/translation unit shown in Fig. 9;

Fig. 11 is a block diagram showing in detail the instruction word memory means in Fig. 9;

Fig. 12 is a schematic diagram of the control circuit shown in Fig. 11;

Fig. 13 is a block diagram showing another embodiment of the instruction word memory means in Fig. 9;

Fig. 14 is a diagram showing the control status of data input/output control on the first-in-first-out basis for the instruction word memory means in Fig. 13;

Fig. 15 is a schematic diagram showing another embodiment of the memory for temporarily storing instruction words in the instruction word memory means in Fig. 11;

Fig. 16 is a schematic diagram of the control circuit for controlling the input/output of the memory shown in Fig. 15;

Fig. 17 is a block diagram showing an embodiment of the control circuit in the instruction word memory means in Fig. 11 for setting the head of data always at the high-order bit for the case where data in the instruction word temporary memory is handled in half-word length;

Fig. 18 is a schematic diagram showing part of the flip-flop in Fig. 17:

Fig. 19 is a timing chart showing the operation of the flip-flop shown in Fig. 18;

Fig. 20 is a schematic diagram showing an embodiment of the code reference means, code arrangement information memory means and arrangement means shown in Figs. 8 and 9;

Fig. 21 is a schematic diagram showing another embodiment of the arrangement means in Fig. 20;

Fig. 22 is a schematic diagram of the code reference means in Fig. 8 constructed using read/write memory cells so that its contents can be translated dynamically;

Fig. 23 is a schematic diagram showing an example of the memory cell in Fig. 22;

Fig. 24 is a block diagram showing part of the processor in Fig. 1 pertaining to the multiple register system;

Fig. 25 is a block diagram of the task number specifying section in the arrangement of Fig. 24;

Fig. 26 is a block diagram of the address generating section in the arrangement of Fig. 24;

Fig. 27 is a set of diagrams used to explain the overlapping states of the general-purpose registers;

Fig. 28 is another diagram used to explain the overlapping states of the general-purpose registers;

Fig. 29 is a block diagram of the comparator in the address generating section in Fig. 26;

Fig. 30 is a flowchart showing the determining process carried out by the area determination circuit in Fig. 26;

Fig. 31 is a diagram used to explain the physical address generating procedure for the main memory address generating circuit in Fig. 26;

Fig. 32 is a block diagram showing the relationship between the main memory and the off-chip main memory;

Fig. 33 is a block diagram of the task boundary detecting circuit;

Fig. 34 is a block diagram showing an embodiment of the microprogram memory/control unit in the arrangement of Fig. 1;

Fig. 35 is a schematic diagram showing an embodiment of the microprogram address register, address decoder, microinstruction memory and microinstruction register;

Fig. 36 is a timing chart showing the signals at various portions of the arrangement shown in Fig. 35;

Fig. 37 is a timing chart showing the signals for the case where the microprogram memory/control unit is halted without suspending the basic clock;

Fig. 38 is a block diagram showing the operation control decoder within the decoder unit shown in Fig. 1;

Fig. 39 is a schematic diagram showing an embodiment of the decoder control latch, first and second register control decoders and delay circuit shown in Fig. 38;

Fig. 40 is a timing chart showing the signals of the first register control recoder;

Fig. 41 is a block diagram showing an embodiment of the operation unit shown in Fig. 1;

Figs. 42, 43 and 44 are schematic diagrams of the functional blocks shown in Fig. 41;

Fig. 45 is a block diagram showing an embodiment of the processor incorporating a main memory operative at least for reading;

Fig. 46 is a block diagram showing an embodiment of the processor capable of varying the number of memory access cycles; and

Fig. 47 is a timing chart showing the operations of the processor shown in Fig. 46 in various number of memory access cycles.

## DESCRIPTION OF THE PREFERRED EMBODIMENTS

Fig. 2 shows an embodiment of this invention, which is applied to a microcomputer 10 constructed on a single semiconductor substrate. The arrangement includes a central processing unit 101 as a nucleus, a main memory 102, a peripheral circuit 103 having the timer function and input/output function, a direct memory access controller 104, and an address translator 105. A processor section 20 enclosed by the dashed line in Fig. 2 will be described in detail in connection with Figs. 1 and 3.

Fig. 1 shows an embodiment of the processor section 20, and it consists of an instruction word memory/translation unit 201, a microprogram memory/control unit 202 including a microprogram ROM (Read Only Memory) as a principal element, a decoder unit 203, an operation unit 204, a bus interface unit 205, a first main memory 206 operative for reading and writing, a second memory 207 operative for only reading, an address input/output buffer 2082, a data input/output buffer 2083, a clock and power supply buffer 2084, and an input/output buffer 2081 for signals, except the address/data clock and power supply, transmitted or received by the processor section 20 to/from the other functional blocks 103, 104 and 105 in Fig. 2.

The following describes the processor operation from instruction fetching up to instruction execution with reference to Fig. 1.

(1) Instruction fetching

The contents of the program (instruction) fetching register (instruction address register) in the bus interface unit 205 are read out over the bus 121 into the main memories 206 and 207 in the processor 20 and, at the same time, into the address input/output buffer 2082 so that the contents are sent out over the bus 222 outside of the processor 20. An instruction word is read out of the first main memory 206 or second main memory 207 by being addressed by the above input onto the bus 213, or an instruction word on the bus 223 is read in through the data input/output buffer 2083 onto the bus 213. Information on the bus 213 is fed into the instruction word memory/translation unit 201.

(2) Instruction latching and translation

The instruction word received by the instruction memory/translation unit 201 is held in it and at the same time decoded into information to be sent to the microprogram memory/control unit 202.

(3) Microinstruction read-out

The microprogram memory/control unit 202 receives the information provided by the instruction word memory/translation unit 201 over the bus 231, and sends it as a microinstruction string over the bus 232.

(4) Microinstruction decoding

The microinstruction from the microprogram memory/control unit 202 is sent over the bus 232 into the decoder unit 203, which decodes the microinstruction into signals 233 which directly control the operation unit 204 and signals 234 which directly control the bus interface unit 205.

(5) Operation execution

The operation unit 204 implements data operations specified by the signals 233. The bus interface unit 205 performs the timing control for data transfer within the processor 20 or between the processor 20 and other functional blocks using the bus 122, input/output buffer 2081, bus 221 and signal lines 235 and 236.

The clock and power voltage supplied to the buffer 2084 are distributed to the units using a signal line 214.

Fig. 3 shows an example of the arrangement of the instruction word memory/translation unit 201, microprogram memory/control unit 202, decoder unit 203 and operation unit 204. The instruction word transferred over the bus 213 to the instruction word memory/translation unit 201 is held temporarily in the instruction word memory 301 and then transferred to the first instruction decoding means, i.e., the instruction decoder 302. Fig. 4 shows typical instruction formats of instruction words read into the instruction memory 301.

(1) Operational instructions

An instruction of this type implements an operation between a first operand specified by the first operand code and a second operand specified by the second operand code and stores the result in the location of the second operand. The type of operation specified in the operation code designates one of the arithmetic or logic operations.

(2) Branching instructions

An instruction of this type causes the program sequence to branch to the program address indicated by an operand specified by the first operand code upon fulfillment of the branch condition given in the operation code. If the branch condition is not met, the instruction causes the program sequence to proceed to the next instruction word.

(3) Bit manipulating instructions

An instruction of this type tests the value of a bit position specified by the bit number, i.e., a first operand, in the second operand specified by a second operand code, and memorizes the bit value. Thereafter, the instruction implements one of operations (clear, set, change, test) indicated by the bit operation type in the operation code for the specified bit, and stores the result in the corresponding bit position of the second operand when necessary.

The instructions categorized in (1), (2) and (3) are systematized at the sacrifice of orthogonality due to the restriction in word length of an instruction as a result of formatting for the intended functions in a limited word length in accordance with a certain rule.

The embodiment shown in Fig. 3 operates in the instruction decoder 302 to decode instruction words read out of the instruction word memory 301 so that information of various kinds necessary for the instruction execution is converted into instruc-

tions having a complete orthogonal code system - (will be termed "intermediate machine words" hereinafter). Each intermediate machine word has an operation code for specifying the operation to be executed and information bits representing some operand specifier codes which identifies the location in the main memory where the operand is contained.

The intermediate machine word is stored in the memory 303, and then fed to the microprogram memory/control unit 202 and the second instruction decoding means, i.e., the decoder unit 203. Some of intermediate machine words include at least one operand specifier, and information of as to whether the operand indicated by the operand specifier is a source operand or a destination operand is entirely expressed in the operation code. Among information expressing the operation code and operand specifier, information of various kinds about the intermediate machine word except for the information 3031 that needs to be decoded by the microprogram memory/control unit 202 is in one-to-one correspondence to the decoders 2031, 2032, 2033, 2034 and 2035 in the decoder unit 203 which can be controlled by the microinstruction 232 (see Fig. 3). Control signals 233 produced by these decoders in the decoder unit 203 control the operation unit 204 so as to execute an intended operation indicated by the intermediate machine word.

Fig. 5 shows another embodiment of the instruction word memory/translation unit, in which there are two instruction decoders for decoding an instruction held in the instruction memory means 301 into an intermediate machine word. In case the main memory storing user programs stores instruction words expressed in two different kinds of instruction systems A and B, i.e., different instruction process (instruction functions) for the same binary instruction code, the system is provided with an instruction decoder 302 for decoding instructions in instruction system A and an instruction decoder 502 for decoding instructions in instruction system B. Information created by the instruction decoder 302 is sent over the bus 3021 to the multiplexer 503. Similarly, information created by the instruction decoder 502 is sent over the bus 5021 to the multiplexer 503. The multiplexer 503 selects one of the buses 3021 and 5021 basing on the contents of the flip-flop 504, and stores the selected contents in the memory 303. In this case, if the instructions expressed in the instruction code of instruction system A and expressed in the instruction code of instruction system B have the same process even though their binary codes are different, the two instruction decoders 302 and 502 generate exactly the same intermediate machine word. Namely, if instruction words of different instruction systems have the same instruction function, they have the same information stored in the memory 303, the most hardware 500 in the processor 20 controlled by the intermediate machine word (hardware in the processor 20 excluding the instruction memory/translation unit 201 in Fig. 2) can be used and controlled commonly by instruction words of different instruction systems.

The flip-flop 504 is set to the predetermined signal level which is received at the terminal 505 and detected by the control circuit 506 when the processor 20 is reset. The terminal 505 uses the lowest-order bit of the address bus.

Fig. 6 shows still another embodiment of the instruction word memory/translation unit 201, in which instruction words read into the instruction memory 301 through the bus 213 and intermediate machine words created by the instruction decoder 302 and instruction decoder 502 and stored in the memory 303, shown in Fig. 5, are stored together in the main memory 600. The main memory stores instruction codes of two instruction systems, one type of codes being fed via the instruction memory 301 to the instruction decoder 302, translated into intermediate machine words by the decoder 302, fed over the bus 3021 to the multiplexer 603, selected by the multiplexer 603 and stored in the memory 303, while another type of codes being fed through the instruction memory 601 and bus 6011 to the multiplexer 603, selected by it and stored in the memory 303. The multiplexer 603 selects one of information on the bus 3021 and bus 6011 depending on the contents of the flip-flop 604. The instruction words stored in the main memory 600 have the definition of the instruction which can switch the multiplexer 603, and by executing the instruction the flip-flop 604 can be controlled through the control line 6041. The flip-flop 604 may be controlled by detecting the signal level received at the terminal 605 by the control circuit 606 as shown in Fig. 7. The multiplexer 603 and memory 303 are designed to control the contents of the instruction memory 601 in byte (8 bits) units, allowing applications which do not need the instruction memory 601 to enter information to the multiplexer 603 in the same word length as of the bus 213.

Fig. 8 is a block diagram showing separately the functions of the instruction memory/translation unit 201 shown in Fig. 3. The functions of the blocks are as follows.

(1) Instruction memory means 301

This is a buffer memory for temporarily holding an instruction word read out from the main memory over the bus 213. An instruction word is read out of the main memory in advance so as to allow high-speed pipeline processing.

(2) Code collation means 802

The arrangement of the operation code and operand code differs depending on each instruction word. On this account, an instruction word read out of the instruction memory means 301 over the bus 3011 is collated with the code pattern which has been defined in the code collation means 802 so that the coincidence is detected.

(3) Code arrangement information memory means 803

Code arrangement information for translating an instruction word into an intermediate machine word having the orthogonal instruction format is defined and stored in this memory means. The defined arrangement information is read out of the memory means over the bus 8031 in response to the signal 8021 from the code collation means 802.

(4) Arrangement means 804

In order to translate an instruction code entered through the bus 3011 into an intermediate machine word having the orthogonal instruction format, arrangement of the instruction code is carried out in accordance with information on the bus 8031. After translation, the intermediate machine word has its operation code for specifying the operation to be executed and its operand specifier for identifying the memory location containing the operand being arranged in separate bit positions. Some instructions except those which do not need operands include at least one operand specifier.

Fig. 9 shows an embodiment of the instruction memory/translation unit shown in Fig. 8, and it consists of an instruction memory means 301 made up mainly of a first-in-first-out (will be termed simply "FIFO" hereinafter) memory 901 and its control circuit 911, a code collation means 802 and code arrangement information means 803 made up mainly of programmable logic arrays (will be termed simply "PLA" hereinafter), and an arrangement means 804 made up mainly of a switch array using, for example, metal oxide semiconductor (will be termed "MOS" hereinafter) transistors. Fig. 10 shows the timing relationship of signals a through j in Fig. 9 together with the basic clocks $\phi_1$ and $\phi_2$.

Initially, an instruction word read out of the main memory is fed over the bus a and latched in the data buffer b. The latched data is transferred over the bus 213 to the FIFO memory 901 in synchronism with the basic clock $\phi_2$. At this time, the control circuit 911 controls the writing of data on the bus 213 into the FIFO memory through the signal line 9110. At the same time, an instruction to be executed is read out of the FIFO memory 901 onto the bus c and it is latched in the FIFO buffer 921. The code collation means 802 and code arrangement information memory means 803 are formed of dynamic PLAs 902 and 903 made solely of NMOS transistors, except for the precharging circuit and driver. On this account, an input latch 912 and output latches 913 and 914 are added. The timing of signals after the signal d has been entered to the latch 912 until the output h of the latch 913 and the output i of the latch 914 are determined is as shown in Fig. 10. The signal h is the instruction code prior to arrangement, and the signal i is arranged information. The arrangement means 804 produces a signal j by making arrangement for the signal h in accordance with the signal i.

Fig. 11 shows in detail the instruction memory means in Fig. 9. The FIFO memory 901 is formed of a single-port random access memory (will be termed simply "RAM" hereinafter). The data lines 1100 are charged by the precharging circuit during the period of the basic clock $\phi_1$, and they carry data during the period of the basic clock $\phi_2$. Sense amplifiers are provided for the amplification of read-out data. The control circuit 911 consists of a write pointer (shown by WP in Fig. 11) for specifying the write location in writing an instruction word on the bus 213 into the FIFO memory 901, a read pointer (shown by RP in Fig. 11) for specifying the read location in reading out an instruction word from the FIFO memory 901, and +1 adders for incrementing the pointers.

Fig. 12 is a logic circuit diagram showing in detail the control circuit 911 shown in Fig. 11. The figure shows an embodiment of the case where the FIFO memory 901 has a capacity of eight words (one word has the same word length as of the bus 213), i.e., each of the write pointer and read pointer is of three bits. A bus 9111 is provided for transferring the contents of the write pointer and read pointer. Both pointers can be reset by the $\overline{\text{RESET}}$ signal which becomes active when the FIFO memory 901 is cleared by

the reset operation for the processor 20 or when a prefetched instruction in the FIFO memory is invalidated due to the execution of a branch instruction. Signal 1200 is the signal indicating that there is no memory space in the FIFO memory 901 for fetching an instruction from the main memory, i.e., the FIFO memory 901 is filled with a prefetched instruction word and there is no space physically. Signal 1201 is the signal indicating that there is a memory space of eight words available in the FIFO memory 901 for fetching instructions from the main memory, i.e., there is no instruction word prefetched in the FIFO memory 901. Signal 1202 is the incrementing signal for the write pointer, signal 1203 is the incrementing signal for the read pointer, signal 1204 is the signal outputting the value of the write pointer onto the bus 9111, and signal 1205 is the signal outputting the value of the read pointer onto the bus 9111.

Fig. 13 shows another embodiment of the instruction memory means 301 in Fig. 9, in which the FIFO memory 901 is controlled by the PLA. The arrangement implements concurrently the control for fetching an instruction word from the main memory to the FIFO memory through the 32-bit bus and the control for transferring the contents of the FIFO memory to the instruction decoder in 16-bit word length. The PLA which controls the data input/output to/from the FIFO memory satisfies the FIFO control state diagram shown in Fig. 14. The PLA 1300 changes in its state in the range of ten states from $S_1$ to $S_{10}$ as shown in Fig. 14, and it becomes the $S_1$ state from any state by receiving the FIFO reset signal. The states $S_1$ and $S_6$ indicate that the FIFO memory 1301 is empty (indicated by E in Fig. 14), i.e., no instruction word is yet fetched to the FIFO memory. The states $S_3$, $S_5$, $S_7$ and $S_{10}$ indicate that the FIFO memory is full (indicated by F in Fig. 14) and there is no space in the FIFO memory 1301 for fetching a new instruction word. The state of the PLA ($S_1$-$S_{10}$) is memorized in the state memory 1302, which is revised by the signal 13002 when the PLA 1300 issues a new state.

Fig. 15 shows another embodiment of the FIFO memory 901 shown in Fig. 11, in which the FIFO memory 901 is controlled for reading and writing through the provision of the input data word length (number of bits of the bus 213) twice the output data word length. The arrangement includes two word lines (15001 and 15002) for the read/write control of memory cells 1110.

Fig. 16 is a detailed logic circuit diagram of the control circuit 1510 for controlling the input/output operation of the FIFO memory 1500 in Fig. 15. A feature of this arrangement as compared with Fig. 12 is the presence of two read pointers ($RP_H$ and $RP_L$), which allows the handling of data in the FIFO memory in both half-word length and full-word length.

Fig. 17 shows as an example the arrangement of the control circuit for setting the head of data always at the high-order bit in handling data in the FIFO memory 1500 in half-word length. Initially, the flip-flop 1700 is reset by the signal 17001 at a time point when the FIFO memory 1500 and control circuit 1510 are reset (state 0). Thereafter, the flip-flop 1700 selects the latch 1702 for the output destination. In case data in the FIFO memory 1500 is treated in half-word length, the flip-flop 1700 is set (state 1) by the signal 17002. At this time, the flip-flop selects the latch 1701 for the output destination. In case data in the FIFO memory 1500 is treated in half-word length, the flip-flop 1700 is reset (state 0) by the signal 17002. By repeating these operations the head of data can be known.

Fig. 18 shows in detail part of the flip-flop shown in Fig. 17, and Fig. 19 shows its operation on a timing chart.

Fig. 20 shows an embodiment of the code collation means 802, code arrangement information memory means 803 and arrangement means 804 shown in Figs. 8 and 9. Signal lines d, e, f, g, h, i and j correspond to the signals referred to by the same symbols in Fig. 9, and their timing relationship is shown in Fig. 10. The code collation means 802 is formed of an AND-type dynamic PLA, while the code arrangement information memory means 803 is formed of an OR-type dynamic PLA. The arrangement means 804 is made of n-channel MOS transistors as shown in the figure. The most noticeable feature of the arrangement is that instead of generating signals to be input to the microprogram memory/control unit 202 and decoder unit 203, i.e., information j to be stored in the memory 303, by decoding an instruction word - (signal line d) to make a new binary code (identical to information j) different completely from the instruction word, information j is created by using part of the binary code expressing the instruction word or completely arranging again the code. Another feature is that for the arrangement of the binary code of an instruction word, arrangement information is defined so that instruction words having the completely same procedure of the arrangement format will fall in the same arrangement even if their codes are completely different. This embodiment realizes the function of the conventional instruction decoder for generating the micro-

program address to be inputted to the microprogram memory/control unit from the instruction word using the hardware structure about 1/5 -1/10 in scale as compared with the conventional decoder.

Fig. 21 shows another embodiment of the arrangement means 804, which differs from the embodiment of Fig. 20 in that information j can be specified arbitrarily in accordance with the output of the OR array of the PLA (the code arrangement information memory means 803). The embodiment of Fig. 20 provides information j by arranging part or whole of the instruction code, whereas the new embodiment can produce a code as information j independent of the instruction code. Namely, information j having an arbitrary code pattern can be obtained for any binary code expressing an instruction word merely by changing the contents of the code arrangement information memory means 803.

Fig. 22 shows an embodiment of the code collation means 802 in Fig. 8, which is constructed using memory cells capable of access for reading and writing so that its contents can be translated dynamically. Information for collating instruction codes is defined word by word sequentially in the memory cells 2210 through the bus 22001. After all bits of the memory cells 2210 have been defined, an instruction word intended for collation is put onto the data lines 22101 through the bus d. The memory cells 2210 compare the defined data with data on the data lines 22101, and drives signal lines with inconsistent results to the low level. If consistent results are reached for all bits of a word, the signal lines 22102 retain the high level. At least one of the signal lines 22102 for coincident detection is used to drive the code arrangement information memory means 803 in the next stage.

Fig. 23 shows in detail an example of the memory cell 2210 in Fig. 22. The memory cell 2210 is written data by being controlled through the signal line W. The data lines D and $\overline{D}$ have opposite polarities, and both lines are driven to the high level during the precharging cycle. At this time, the signal line C is driven to the low level, and therefore the precharging operation is desirable for the code arrangement information memory means 803.

Fig. 24 shows part of the processor shown in Fig. 1 pertinent to the multiple register system. In this arrangement, the bus interface unit 205 in Fig. 1 is split into a bus interface unit 2101 and another bus interface unit 2102, while the instruction word translation unit 201, microprogram memory/control unit 202 and decoder unit 203 are shown as a unitary control unit 2000. The bus input/output units 2081, 2082 and 2083 in Fig. 1 are shown as a bus input/output unit 2103. The bus interface unit 2101 includes an address generator 2104, task number specifier 2105, task number memory 2106, base address specifier 2107 and task boundary detection circuit 2108, all controlled by the control unit 2000. Among these functional blocks, the bus interface unit 2102, address generator 2104, task number memory 2106 and base address specifier 2107 are connected to the D-bus 1a and A-bus 1b, while the main memory 102, operation unit 204 and task number specifier 2105 are connected to the D-bus 1a. The control unit 2000 provides a control signal 1c to the operation unit 204, the control signal 1d to the bus interface unit 2102, a control signal 1e and register specifying information 1g included in the signal 1f from the bus interface unit to the address generator 2104, a control signal 1h to the task number specifier 2105, a control signal 1i to the taks number memory 2106, and a control signal j to the base address specifier 2107. The task number specifier 2105 is designed to receive the event occurrence signal 1k in response to an interrupt request from a peripheral unit (not shown) and the event occurrence signal 1ℓ at the time of procedure call/return. The operation unit 204 is connected with the bus interface unit 2102 through a data transfer bus 1m. The signal 1n carries the task number obtained in the task number specifier 2105, and the signal 1p carries physical address information issued by the address generator 2104 to the main memory 102.

The following describes in brief the operation of the processor 20 incorporating the foregoing functional blocks. An instruction word read out of the main memory outside of the processor 20 is fed over the D-bus 1a to the bus interface unit 2102. The instruction word is also transferred in some cases to the control unit 2000 as signal 1f. The control unit 2000 decodes the input instruction word and, if it is an instruction using a register, sends register specifying information which is set in the register specifying field in the instruction word to the address generator 2104 and the control signal 1c to the operation unit 204.

The task number specifier 2105 responds to the event occurrence signal 1k or 1ℓ, which arise in response to an external interrupt request or the execution of an internal subroutine interrupt, to select a task corresponding to the event and at the same time issues a task number of identifying a register set corresponding to the task to the specified address generator 2104 in accordance with the procedure described later. The address generator 2104 generates a physical address corresponding to the main memory 102 from the task number 1n and register specifying information 1g in accordance with the procedure described later, and

sends it to the main memory 102. The main memory 102 reads out the contents of the location corresponding to the given physical address, and sents it over the D-bus 1a. The operation unit 204 takes data on the D-bus 1a in response to the control signal 1c, and implements the specified operation. After the operation, the resultant data is fed over the D-bus 1a and written into the memory specified by the instruction.

If, during the execution of a task, an event occurrence signal 1k or 1ℓ higher in priority than the task is given, the task number specifier 2105 issues a task change request by the signal 1q, and the control unit 2000 operates to save various data such as the task number pertinent to the task in execution through the D-bus 1a to the stack in the main memory. The control unit 2000 further implements the control so as to modify the environment to meet the task of the higher priority event. In the same way as the previous case, the task number specifier 2105 issues a task number which identifies a register set corresponding to the new task. In this way, each time an event occurrence signal 1k or 1ℓ has been applied its priority is compared with that of the current task and the task is replaced when the latter event is higher in priority. When the task number 1n has exceeded the number of tasks 1r preset in the task number memory 2106, it is detected by the task boundary detection circuit 2108 and indicated to the control unit by the signal 1s so that the control unit performs the predetermined process.

Fig. 25 shows the detailed arrangement of the task number specifier 2105, which consists of a first flip-flop group 2111 and second flip-flop group 2112 for timing the event occurrence signals 1k, a priority arbitration circuit 2113 for testing the priority among the events, a task number register 2114 for holding the task number, an adder/subtracter 2115 for incrementing or decrementing by one the contents of the task number register 2114 depending on the value 1h-3 which is part of the control signal 1h, and a task number latch 2116 for temporarily holding the task number.

The operation of the above arrangement will be described with reference to Fig. 25. The event occurrence signals 1k entered asynchronously with the basic clocks $\phi_1$ and $\phi_2$ in the processor 20 are sampled by the first flip-flop group 2111 and synchronized with the basic clock $\phi_2$ by the second flip-flop group 2112. Synchronized event signals are delivered to the priority arbitration circuit 2113, and an event signal which is determined to be higher in priority than the current task causes the issuance of the event occurrence detect signal 1q to the control unit 2000. In response to this signal, the control unit 2000 issues a control signal 1h to replace the task number which has been effective until the event has occurred with the new task number in the task number register 2114. In case a procedure call/return or the like occurs during the task process execution, the control unit 2000 issues a signal 1ℓ so as to replace the task immediately.

It should be noted that the task number register 2114 is connected to the D-bus 1a with the intention of allowing the instruction to access to the task number register 2114 for reading or revising the contents. In this embodiment, when the instruction for revising the contents of the task number register 2114 has been executed, the task number indicated by the task number register 2114 immediately before the execution of the instruction, i.e., the task number before revision, is saved to the stack using the stack pointer in the register set specified by the revised task number after the execution of the instruction. In order to retrieve the former task number which has been saved, there is provided an instruction which reads out the contents of the stack indicated by the stack pointer in the register set specified by the current task number and sets it in the task number register 2114.

Fig. 26 shows in detail the arrangement of the address generator 2104. The address generator 2104 is to produce physical address information for the main memory 2107 basing on the task number 1n and register specifying information 1g, and it consists of a global base register 2121, a local base register 2122, a comparator 2123, an area determination circuit 2124 and a RAM physical address generating circuit 2125. The global base register 2121 and local base register 2122 can be accessed by the instruction for revising the contents. The comparator 2123 operates to compare the contents 3a of the global base register 2121 and the register number included in the register designating information 1g and also compare the contents 3b of the local base register 2122 with the register number in 1g. The results of comparison are delivered as signals G and L, respectively, to the area determination circuit 2124. The area determination circuit 2124 determines one of the global area, local common area and local bank area in accordance with the signals G and L, and provides information as a control signal 3c necessary for generating the physical address of the main memory 102. The RAM physical address generating circuit 2125 produces the physical address of the on-chip RAM from the task number 1n and register number included in the register specifying information 1g in accordance with the control signals 3c

and 1e. In this embodiment, the main memory 102 has a capacity of 1152 bytes, 64 bytes for each task. Accordingly, sixteen 32-bit registers can be provided for each task.

The following describes, as an example, the operation of each functional block of the address generator.

(1) Task switching from an even-numbered task to an odd-numbered task

When the global base register 2121 contains "4" and the local base register 2122 contains "4", the registers for task #2n+1 are grouped into a global area, local common area and local bank area, as shown in Fig. 27(a). These areas include registers as follows.

(i) Global area : R0-R3

(ii) Local common area: R15-R12

(iii) Local bank area : R4-R11

(2) Task switching from an odd-numbered task to an even-numbered task

When the global base register 2121 contains "4" and the local base register 2122 contains "4', the registers for task #2n+2 are grouped into the three areas as in case (1). These areas include registers as follows.

(i) Global area : R0-R3

(ii) Local common area : R4-R7

(iii) Local bank area : R8-R15

Supposing a task switching from task #1 to task #2, which is relevant to the above case (2) of task switching from odd to even task number, the following operation takes place.

(i) Access to registers R0-R3 in the global area

When registers R0-R3 for task #2 are specified, R0-R3 for task #0 are accessed.

(ii) Access to registers R4-R7 in the local common area

When registers R4-R7 for task #2 are specified, R4-R7 for task #1 are accessed.

(iii) Access to registers R8-R15 in the local bank area

When registers R4-R7 for task #2 are specified, R4-R7 for the current task #2 are accessed.

These access operations are shown in Fig. 28. In any task, when a register in the global area is specified, a corresponding register for task #0 is accessed. When a register in the local common area is specified, a corresponding register for the previous task is accessed. When a register in the local bank area is specified, a corresponding register for the current task is accessed.

Accordingly, when it is intended to transact parameters among tasks, the only requirement is to specify the contents of the global base register 2121 and local base register 2122 in advance and actual parameter transaction among tasks is not needed, whereby high-speed task switching is made possible.

Fig. 29 shows in detail the arrangement of the comparator 2123. The comparator 2123 consists of a comparison circuit 2131 capable of identifying a global area and a comparison circuit 2132 capable of identifying a local bank area. The comparison circuit 2131 compares the register number A included in the register specifying information 1g with the global base value B which is the output 3a of the global base register 2121, and produces an output signal G at logical level "1" if A is smaller than B, or otherwise produces a "0" output. The comparison circuit 2132 consists of a comparison circuit 2133 capable of identifying a local bank area when switching has been made from an odd-numbered task to an even-numbered task, and a comparison circuit capable of identifying a local bank area when switching has been made from an even-numbered task to an odd-numbered task, both circuits having their outputs 6a and 6b at logical level "1" when the register number A belongs to the local bank area. A multiplexer 2136 selects one of the outputs 6a and 6b to produce a signal L in accordance with the output 6 of the flip-flop 2135 which is controlled by the signal 1e from the control unit 2000.

Fig. 30 shows the process carried out by the area determination circuit 2124.

Initially, the relevance of the global area is tested basing on the value of the signal G.

(i) Affirmative test result: The circuit recognizes the area to be the global area, and issues a signal 7a dictating that the physical address of the main memory 102 be generated using the global task number and register number.

(ii) Negative test result: The circuit recognizes the area to be not the global area, and proceeds to the next decision step. In this step, the circuit tests whether the area is the local bank area or the local common area basing on the value of the signal L.

(iii) Affirmative test result: The circuit recognizes the area to be the local bank area, and issues a signal 7c dictating that the physical address of the main memory 102 be generated using the current task number and register number.

(iv) Negative test result: The circuit recognizes the area to be the local common area, and issues a signal 7b dictating that the physical address of the main memory 102 be generated using the previous task number and register number.

These signals 7a, 7b and 7c produced as a result of the above decision process are delivered as signal 3c to the RAM physical address generating circuit 2125.

Fig. 31 shows the procedure of physical address generation by the main memory address generating circuit 2125. In this embodiment, the main memory 102 has a capacity of 1152 bytes and therefore the physical address ranges from 0 to 1151. The following describes the operation of the case of 16 tasks provided in the main memory 102. In this case, the low-order four bits of the task number 1n provided by the task number specifier 2105 are effective to indicate the task number. The physical address is generated in accordance with the following procedure.

(1) Step 1

The address generator receives a task number $(t3t2t1t0)_2$ which is based on the register number $(r3r2r1r0)_2$ included in the register specifying information 1g and the signal 3c provided by the area determination circuit 2124.

(2) Step 2

The task number is complemented to obtain $T1=(\bar{t}3\ \bar{t}2\ \bar{t}1\ \bar{t}0)_2$.

(3) Step 3

The T1 shifted to the left with a "0" being entered to the lowest-order bit. The shift operation is repeated six times to obtain $T2=(\bar{t}3\ \bar{t}2\ \bar{t}1\ \bar{t}0\ 0\ 0\ 0\ 0\ 0\ 0)_2$.

(4) Step 4

The register number is shifted to the left with a "0" being entered to the lowest-order bit. The shift operation is repeated twice to obtain $R1=(r3r2r1r0\ 0\ 0)_2$.

(5) Step 5

The T2 is added to the R1 to obtain $A1=(t3t2t1t0r3r2r1r0\ 0\ 0)_2$.

(6) Step 6

Offset value $(080)_{16}$ is added to the A1 to obtain the result A2. For example, task number $(0000)_2$ produces the result $A2=(1\ 0\ 0\ 0\ 1\ r3r2r1r0\ 0\ 0)_2$. The offset value at this time is determined uniquely from the total capacity of the main memory 102, and it takes the above-mentioned value for a 1152 byte main memory.

(7) Step 7

The main memory 102 is accessed with high-order nine bits "1 0 0 1 r3r2r1r0" of A2 being set to the high-order nine bits of the address. Namely, four bytes are accessed concurrently. This operation results from the assumption of 32-bit registers, and is confined to this embodiment.

Fig. 32 shows the relationship between the off-chip main memory 2142 and on-chip main memory 102 based on the contents of the main memory base register 2141 included in the base address specifier 2107. The main memory base register 2142 can be accessed by the instruction for setting the contents arbitrarily. When an arbitrary address of the off-chip main memory 2142 is set in the main memory base register 2142, an address space ranging from that address up to 1152 is assigned to the on-chip main memory 102. Accordingly, when the address of the memory specified by the bus interface unit 2102 is included in the above-mentioned address space, the processor 20 makes access to the main memory 102. Namely, it is controlled so that the main memory 102 in the processor 20 is arranged in arbitrary location of the off-chip main memory 2142 by merely changing the contents of the main memory base register 2141. This embodiment is provided with an address detection means for distinguishing the register area in the main memory 102 and other area so that the area defined to be register sets for the task process is not accessed as a memory area.

Fig. 33 shows the arrangement of the task boundary detection circuit 2108. The task number 1n is compared with the number of tasks 1r which is set in the task number memory 2106, and if the task number 1n has exceeded the number of tasks 1r, it is indicated by the signal 1s to the interrupt vector generating circuit 2143 in the control unit 2000, which then executes a predetermined exceptional process. In this embodiment, the task number memory 2106 can be accessed by the instruction for revision, and the memory contents signifies the number of tasks as follows.

(i) 0: two tasks

(ii) 1: four tasks

(iii) 2: eight tasks

(iv) 3: 16 tasks Accordingly, for the task number memory contents of N, the maximum task number is $2^{N+1}$ . There is provided a maximum value generating circuit 2144 for calculating $2^{N+1}$ form N, but the provision of this circuit 2144 is confined to this embodiment.

Fig. 34 shows an embodiment of the microprogram memory/control unit 202. Part of the microprogram address to be set in the microprogram address register 2403 is selected by the selector 2402. The selected signal 2402a is merged with the signal 2406e which is part of the output from the microinstruction register 2406, and the result is held in the microprogram address register 2403. The output of the microprogram address register 2403 is delivered to the address decoder 2404 and, at the same time, incremented by the +1 adder 2408 and the resultant signal 2408a is fed to the selector 2402. The address decoder 2404 decodes the microprogram address and sends the result to the microinstruction memory 2405. The microinstruction memory 2405 reads out a specified microinstruction and sets it in the microinstruction register 2406. The microinstruction register 2406 provides three major outputs including signal 2406a which is the signal for controlling the decoder unit 203 and operation unit 204, signal 2406b which is the signal for specifying a branch destination address when the microprogram makes branching, and signal 2406c which is the signal for controlling the next address of the microprogram. A control circuit 2407 responds to the signal 2406c to produce a signal 2407a for controlling the selector 2402 and a signal 2407b for controlling the stack 2409 used for saving the microprogram address.

Fig. 35(a) shows an embodiment of the microprogram address register 2403, address decoder 2404, microinstruction memory 2405 and microinstruction register 2406 shown in Fig. 34. Each bit of the microprogram address register 2403 is formed of a flip-flop 2503 as shown in Fig. 35(b), for example. The address decoder 2404 is formed of an AND-type dynamic PLA, while the microinstruction memory 2405 is formed of an OR type dynamic PLA. The microinstruction register 2406 is formed of a dynamic latch.

Fig. 36 shows the timing relationship of the signals together with the basic clocks $\phi_1$ and $\phi_2$. As can be seen from Fig. 36, one clock period is needed to obtain the output of the microinstruction register 2406 after the output 2403a of the microprogram address register 2403 has settled.

Fig. 37 shows the timing relationship of the signals when the microprogram memory/control unit 202 is halted without suspending the basic clocks $\phi_1$ and $\phi_2$. Signal 24A is the signal for halting the microprogram memory/control unit 202, and the signal at high level causes the microprogram address register 2403 to stay unchanged and the microinstruction register 2406 to produce a low level output for all bits. By assigning this special value (all low bits) of the microinstruction register to be a "no operation" command for the application system, the microprogram memory/control unit 202 can apparently be halted. The example of Fig. 37 shows the timing relationship of the case where a halt of one clock period is placed (shown by Nop in the figure) between the cycle of microinstruction 2 (shown by $\mu$IR2 in the figure) and the cycle of microinstruction 3 (shown by $\mu$IR3 in the figure).

Fig. 38 shows the arrangement of the operation control decoder 2030 in the decoder unit 203. The decoder 2030 consists of a decoder control latch 2800, a first register control decoder 2810, a second register control decoder 2820, an operation circuit control decoder 2840 and delay circuits 2830 and 2850. The first register control decoder 2810 specifies the source register which contains data to be operated, the second register control decoder 2820 specifies the destination register in which the operation result is stored, and the operation circuit control decoder 2840 specifies the type of operation (addition, subtraction, logical sum, logical product, exclusive logical sum). It takes one clock period after a source register has been read until the result of operation is reached, and therefore the signal 2801 is delayed for one clock period by the delay circuit 2830. Because of different timing of use of the output signals 2841 and 2842 from the operation circuit control decoder 2840, the signal 2842 is delayed by the delay circuit 2850.

Fig. 39 shows an embodiment of the decoder control latch 2800, first register control decoder 2810, second register control decoder 2820 and delay circuit 2830 shown in Fig. 38. The most noticeable feature of this embodiment is that the first and second register control decoders 2810 and 2820 are arranged by AND-type dynamic PLAs, whose output lines 2812 are precharged using p-channel MOS transistors, a specific level - (high level in this embodiment) of the signals 2812 outputted during the precharging period is defined to be the unselected state for the input/output of the register, only signal lines 2812 which meet the PLA logic following the precharging period are discharged, and the register input/output lines 2813 are made the register selection stage.

Fig. 40 shows the timing relationship of the signals of the first register control decoder 2810 in relation with the basic clocks. It takes one clock period after the output 2801 of the decoder control unit 2800 has settled until the register contents are read out by the register read-out control signal 2811. This embodiment can be realized by hardware as little as about one third of the static PLA structure. Only selected ones of the signal lines 2812 carry discharge currents, which reduces the power dissipation, and the circuit is operative in the register access timing comparable to the conventional static decoder. When logical sum for the output lines of the PLA is required, it can be achieved by the wired-OR configuration for the output lines.

Fig. 41 shows, as an example, the arrangement of the operation unit 204. The unit 204 consists of a temporary register 3100 for holding data, a status register 3110 for holding the state of the operation result, an operation circuit 3120, a source latch 3130 for temporarily holding data to be operated, a shift circuit 3140 operative to shift one bit right or left for the operation result, a destination latch 3150 for temporarily holding the operation result, a read data register 3160 for temporarily holding data read on the bus 213 and a write data register 3170 for temporarily holding data sent out over the bus 213.

Initially, one of operation data is fed over the bus 31A into the source latch 3130. The contents of the source latch 3130 and read data register 3160 are entered to the operation circuit 3120, which implements an operation specified on the signal line 2823. The result of operation is held temporarily in the destination latch 3150, and then it is fed over the bus 31B to a register specified by the signal 2813. Signals produced as a result of operation indicative of the sign of the result, the result of zero, the occurrence of carry (borrow), the occurrence of overflow are fed over the signal lines 31C into the status control circuit 3180, and then the states of the signals specified by the signal 2803 which is part of a microinstruction are stored in the status register 3110.

Figs. 42, 43 and 44 are schematic diagrams showing an example of the functional blocks shown in Fig. 41. The buses 31A and 31B are precharged during a high period of the basic clock $\phi_1$, and they transfer data in a high period of the basic clock $\phi_2$.

(1) Bit arrangement of temporary register 3100

Each bit of the temporary register 3100, e.g., the lowest-order bit, is made up of a write gate 3101-0 connected to the bus 31B-0, a drive gate 3103-0, a feedback gate 3102-0, and a read gate 3104-0 connected to the bus 31A-0. The temporary register is written by making the control signal 3200a high so that data on the write bus 31B-0 is conducted through the write gate 3101-0, and it is read out by making the control signal 3200b high so that the output of the drive gate 3103-0 is conducted through the read gate 3104-0 onto the read bus 31A-0.

(2) Status control circuit 3180 and status register 3110

The status control circuit 3810 is formed of a PLA which receives the microinstruction 2803 and status information 31C reflecting the operation result, and produces the set signal 3281 written in the status register and data 3282 to be written. The status register 3110 is similar to the temporary register 3100, but has an additional input gate 3110-0 as shown for the lowest-order bit. It should be noted that the write operation of data 3282 through the gate 3110-0 takes place during the precharge period for the buses 31A and 31B in the operation unit 204.

(3) Source latch 3130

This is a dynamic latch for temporarily holding operation data, and it can receive data in inverted version in accordance with the control signal 2813.

(4) Operation circuit 3120

The circuit implements the addition, logical sum, logical product and exclusive logical sum operations in accordance with the control signals on three control lines 3321.

(5) Shift circuit 3140

The circuit shifts data by one bit right or left in accordance with the states of the signal lines 3341. A high state of signal line 3341R causes a right shift, while a high state of signal line 3341L causes a left shift.

(6) Destination latch 3150

This is a dynamic latch for temporarily holding the output of the shift circuit 3140, and it is the only source of data to the bus 31B.

(7) Read data register 3160 and write data register 3170

The read data register 3160 holds data on the bus 213 and transfers it to one input of the operation circuit 3120 or onto the bus 31A. The write data register 3170 holds data to be sent over the bus 213, and it receives data on the bus 31B which carries the operation result.

Fig. 44 shows the arrangement of the casw ehere the bus 213 has 16 bits and the read data register 3160 and write data register 3170 have 32 bits each. Control signals 236a, 236b, 236c and 236d for controlling the input/output to the bus 213 operate on the above two registers so that their high-order 16 bits and low-order 16 bits are controlled independently.

Fig. 45 shows an embodiment of the processor 20 incorporating a main memory 3500 capable of access for at least reading. When the processor 20 is reset, a level detect/memory circuit 3531 detects and memorizes the level at terminal 3530. The operation of the processor 20 will be described for the following two cases.

(1) Operation when the level detect/memory circuit 3531 provides a low output 3532

The contents of the memory address register 1 (shown by MAR1 in Fig. 45) are fed over the bus 212 through the buffer 3520 to the terminal 3521. The multiplexer 2 (shown by MPX2 in the figure) 3550 selects the contents on the bus 212 and supplies a selected one to the main memory 3500. In case the value on the bus 212 indicates the address of the main memory outside of the processor 20, the buffer 3510 selects data on the bus 213 received at the terminal 3511. In another case when the bus 212 indicates the address of the main memory within the processor 20, the buffer 3510 selects data on the bus 3501. The buffer 3510 is controlled by the output 3561 of the decoder 3560 which discriminates whether the address carried by the bus 212 indicates the main memory 3500 in the processor 20. The output 3512 of the buffer 3510 is selected by the multiplexer 1 - (shown by MPX1 in the figure) 3540 and sent to the instruction memory means 301.

(i) Instruction fetch

The contents of the instruction address register (IAR) are set in the memory address register 1 - (MAR1) to read out the main memory.

(ii) Data read/write

The contents of the data address register - (DAR) are set in the memory address register 1 - (MAR1) to read or write the main memory. At this time, the data address register (DAR) contains the effective address which has been calculated by the operation unit 204.

(2) Operation when the level detect/memory circuit 3532 provides a high output 3532

(i) Instruction fetch

The contents of the instruction address register (IAR) are set in the memory address register 2 - (MAR2) and transferred over the bus 3502 to the multiplexer 2 (MPX2) 3550. The multiplexer 2 - (MPX2) selects the contents of the bus 3502 and supplies a selected one to the main memory 3500. The output of the main memory 3500 is fed over the bus 3501, selected by the multiplexer 1 - (MPX1), and transferred to the instruction memory means 301.

(ii) Data read/write

The effective address which has been calculated by the operation unit 204 and held in the data address register (DAR) is transferred to the memory address register 1 (MAR1) and transferred over the bus 212 to the buffer 3520. The contents of the buffer 3520 are fed through the terminal 3521 to the main memory outside of the processor 20. Data to be read or written is transacted through the terminal 3511 and bus 213 between the operation unit 204 in the processor 20 and the main memory outside of the processor 20 or the main memory in connection with the bus 213 within the processor 20. Namely, with the level detect/memory circuit 3531 providing a high output 3522, instruction fetching takes place with the main memory 3500

within the processor 20 and data reading/writing takes place with a main memory other than 3500. This concurrent operation for instruction fetching and data reading/writing is the feature of this embodiment, and it reduces the instruction processing time.

Fig. 46 shows an embodiment of the processor 20 which is operative in variable memory access cycles (instruction fetch cycle, data read cycle and data write cycle). The length of the memory access cycles is varied in accordance with the contents of the register 3600 which can be accessed for reading or writing by the instruction.

Fig. 47 shows, as an example, memory access cycles determined by the values of the register 3600. Register contents "0" shown in Fig. 47 (a) achieves the fastest memory access. By constructing the microprogram memory/control unit 202 to suit this case, or by halting the operation for one clock period by issuing a signal 24A from the memory access control circuit to the microprogram memory/control unit 202 for the case of register contents "1", a 3-clock memory access can be realized. Namely, according to this embodiment, the access cycle of the main memory used in combination with the processor 20 can be varied by setting the register using the instruction, whereby the system can be adapted to the variation of access cycle which results from the change in the main memory used, without adding extra control circuits outside of the processor 20.

While particular embodiments of the invention have been shown and described, it will be obvious to those skilled in the art that various changes and modifications may be made without departing from the present invention in its broader aspect.

## Claims

1. A data processing system for processing data in accordance with instruction words comprising:

(A) instruction word memory means (301) for storing at least one instruction word given by one reading operation;

(B) first instruction decoding means (302) connected to said instruction word memory means and adapted to arrange an instruction word read out of said memory means into an intermediate machine word having an orthogonal code;

(C) memory means (303) connected to said first instruction decoding means; and

(D) at least one microprogram memory means - (202) connected to said memory means and at least one second instruction decoding means - (2030-2035) for decoding a microinstruction,

said first instruction decoding means extracting information in one-to-one correspondence to inputs of said second instruction decoding means from said instruction word and storing said information in said memory means, said second instruction decoding means being operative to invalidate outputs of said second instruction decoding means using a microinstruction outputted from said microprogram memory means.

2. A data processing system in which at least one of instruction words including operation information for specifying an operation to be executed is an instruction word including at least one piece of operand specifying information for identifying a location of a main memory (102) at which an operand is contained, said operand being processed in response to said instruction word, said system comprising:

(A) instruction word memory means (301) for sequentially, in general, searching and storing instruction words which are entered successively;

(B) instruction decoding means (302, 502) connected to said instruction word memory means and adapted to detect positions of operation information and operand designating information included in an instruction word and extract the operation information and operand specifying information separately; and

(C) memory means (303) connected to said instruction word memory means and having predetermined bit positions for storing the operation information and operand specifying information,

said system being operative to extract operation information and operand specifying information included in said instruction word, arrange said information in said predetermined bit positions of said memory means, and implement processing in accordance with contents of said memory means.

3. A data processing system having a main memory (102) for storing instruction words which include operation information specifying operations to be executed, at least one of said instruction words being an instruction word including at least one piece of operand specifying information for identifying a location of said main memory contain-

ing an operand, said main memory having contents expressed in instruction words having at least two kinds of instruction formats of different instruction binary code even though instruction functions of said instruction words are equal, said system further having a data processor (20) for processing operand in response to said instruction words having at least two instruction formats, said data processor comprising:

(A) instruction word memory means (301) which sequentially, in general, searchs and stores instruction words resumed successively from said main memory;

(B) a plurality of instruction decoding means (302, 502) connected to said instruction word memory means and adapted to detect positions of operation information and operand specifying information included in an instruction word, and extract operation information and operand specifying information separately, said decoding means being in correspondence to a plurality of instruction formats of instruction words in said main memory;

(C) a multiplexer (503, 603) for selecting one of outputs of said plurality of instruction decoding means;

(D) control means (506) for controlling said multiplexer; and

(E) a memory (303) connected to said multiplexer and having predetermined bit positions for storing operation code information and operand designating information,

said system extracting operation information and operand designating information included in an instruction read out of said main memory, arranging said information in said predetermined bit positions in said memory, and implementing processing in accordance with contents of said memory.

4. A data processing system according to claim 3, wherein said multiplexer control means is adapted to control said control means by executing an instruction in said main memory or an instruction supplied from outside.

5. A data processing system, in which at least one of instruction words including operation information for specifying an operation to be executed is an instruction which includes at least one operand specifying information for identifying a location of a main memory (102) containing an operand, and said system processes said operand in response to said instruction, said system comprising:

(A) instruction memory means (301) for sequentially, in general, searching and storing instruction words entered successively;

(B) instruction decoding means (302) connected to said instruction word memory means and adapted to detect locations of operation information and operand specifying information included in an instruction word and extract operation information and operand specifying information separately;

(C) a memory (303) having predetermined bit positions for storing operation information and operand specifying information;

(D) a first bus (3021) for transferring information which has been extracted and arranged for storing operation information and operand specifying information included in an instruction word read out of said main memory in said predetermined bit positions of said memory;

(E) a second bus (6011) for connecting said memory with said main memory;

(F) a multiplexer (603) operative to select one of said first bus and said second bus; and

(G) a control circuit (604, 606) operative to control said multiplexer, said system implementing process in accordance with information stored in said memory.

6. A data processing system according to claim 5, wherein said multiplexer control means is adapted to control said control means by executing an instruction in said main memory or an instruction supplied from outside.

7. A data processing system according to claim 5, wherein said main memory for storing instructions and operands and said data processor are constructed on a single semiconductor substrate.

8. A data processing system operative to control an operation section (204) thereof by reading out a microinstruction at a relevant address in a microprogram memory (202) in accordance with an instruction word supplied from outside, said system comprising:

(A) instruction word memory means (301) for

sequentially, in general, searching and storing at least one instruction word in instruction words entered successively from outside; and

(B) instruction decoding means (302) connected to said instruction word memory means and adapted to generate addresses of said microprogram memory by decoding said instruction word, said instruction decoding means comprising:

code collation means (802) for collating an instruction code with predetermined information to detect the coincidence;

code arrangement information memory means - (803) which receives a result from said code collation means and defines arrangement information for changing whole or part of the instruction code; and

arranging means (804) connected to said code arrangement information memory means and adapted to arrange the instruction code in accordance with said arrangement information, a result from said arranging means in correspondence to said instruction code being an address of said microprogram memory.

9. A data processing system according to claim 8, wherein at least one of said code collation means and code arrangement information memory means is formed of a programmable logic array (902, 903).

10. A data processing system according to claim 8, wherein said code collation means is formed of a random access read/write memory constructed of memory cells capable of detecting the coincidence of a memorized value with a value on data lines.

11. A data processing system for sequentially executing instruction words read out of a main memory (102) or instruction words supplied from outside, said system being provided with instruction word memory means (1500) for sequentially, in general, searching and storing at least one instruction words in instruction words resumed successively from said main memory, said instruction word memory means comprising a memory section for storing an instruction word and a control section for controlling the data read/write operation for said memory section, said instruction word memory means being capable of reading out data in a data length half a data length used in a single write operation.

12. A data processing system according to claim 11, wherein said memory control section is formed of a programmable logic array.

13. A data processing system of the type of multiple register sets system having an additional main memory (102) for storing instructions and operands and adapted to implement data processing in response to said instructions, said system comprising:

a. task number specifying means (2105) for specifying a task number which identifies arbitrary one of a plurality of register sets;

b. number-of-tasks memory means (2106) for memorizing information capable of identifying said plurality of register sets;

c. a random access read/write memory (102) which forms part or whole of said main memory and has a capacity larger than a memory capacity necessary for forming register sets in number specified by said number-of-tasks memory means; and

d. address generating means (2104) connected with said memory and adapted to produce a physical address of said memory from register specifying information included in an instruction word and said task number,

said system being operative to make access to a memory location indicated by a physical address provided by said address generating means upon confirming that the task number specified by said task number specifying means does not exceed a total number of tasks memorized in said number-of-tasks memory means when said instruction is an instruction having access to a register.

14. A data processing system according to claim 13, wherein said number-of-tasks memory means comprises a memory capable of being addressed by an instruction word, said memory being capable of rewriting through execution of said instruction word.

15. A data processing system according to claim 13, wherein said task number specifying means comprises a task number specifying register - (2114) which can be accessed by an instruction word and can be rewritten through execution of said instruction, said system implementing task process by revising contents of said task number

specifying register by a constant number each time a task switching request occurs.

16. A data processing system according to claim 13 further comprising addressing means (2125) capable of indicating an arbitrary physical address location of said main memory, said system making access to a predetermined corresponding portion of said memory when a continuous memory range from a first physical address to a second physical address different from said first physical address of said main memory indicated by said address specifying means is accessed.

17. A data processing system according to claim 13 further comprising detection means (2108) for detecting that a task number specified by said task number specifying means has exceeded a total number of tasks indicated by information stored in said number-of-tasks memory means.

A data processing system according to claim 13 wherein said address generating means comprises:

a. a first rewritable register (2121, 2122) capable of being specified by an instruction word; and

b. identification means (2123, 2124) capable of identifying a series of registers numbered by one register number through another register number which are set in a register set, said first register having contents capable of specifying one of said two register numbers, said system making access to a register with a corresponding register number in a register set indicated by a task number which has been determined irrespective of a current task number when an instruction has specified a register included in said series of registers during execution of a task.

19. A data processing system according to claim 13, wherein said address generating means comprises:

a. a first rewritable register (2121, 2122) capable of being specified by an instruction word; and

b. identification means (2123, 2124) capable of identifying a series of registers numbered by one register number through another register number which are set in a register set, said first register having contents capable of specifying one of said two register numbers, said system making access to a register with a corresponding register number in a register set indicated by a task number immediately preceding a current task number when a register included in said series of registers has been accessed by an instruction during execution of a task following at least one task change.

20. A microprogram control system operative to read out a microinstruction from a memory which stores a plurality of microinstructions in accordance with information held in a memory address register, store said microinstruction in a microinstruction register, and control an operation section basing on contents of said microinstruction register, said system comprising:

first means which inhibits renewal of said memory address register upon occurrence of a signal for suspending functions of said system; and

second means which generates a specific signal for invalidating a read-out microinstruction so that said operation section performs nothing apparently, whereby said microprogram control system halts its functions apparently without suspension of a basic clock supplied to said memory for storing microinstructions and without halting the read-out operation for said memory.

21. A microprogram control system according to claim 20, wherein said first means for inhibiting renewal of said memory address register comprises a NOR gate which receives a basic clock for renewing said memory address register and a signal for halting functions of said microprogram control system.

22. A microprogram control system according to claim 20, wherein said second means for invalidating a read-out microinstruction so that said operation section performs nothing apparently is operative to inhibit renewal of said microinstruction register and making whole output of said microinstruction register to be a low level invariably.

23. A microprogram control system according to claim 20, wherein said second means for invalidating a read-out microinstruction so that said operation section performs nothing apparently comprises a logic circuit connected to the output of said microinstruction register, said logic circuit being provided with a function of logical product or logical sum for a signal from said microinstruction register and a signal for halting functions of said microprogram control system so that the input from said microinstruction register

is invalidated by the signal for halting functions of said microprogram control system.

24. A data processing system adapted to control its operation section in such a way of reading out a microinstruction from a microprogram memory addressed by an instruction read out of a main memory or an instruction supplied from outside, decoding said instruction and producing a control signal for said operation section, said system comprising a register control decoder for producing a control signal for controlling read/write operations of registers within said operation section in accordance with a microinstruction, said decoder being formed of an AND-type dynamic programmable logic array, said array having output lines precharged by use of p-channel MOS transistors, said register being controlled for reading and writing by a signal outputted during a precharge period with a first specific level of said signal being defined to be an unselected state for input/output of said register, and with a second specific level of said signal being defined to be selected state for input/output of said register at a time of discharging for only output lines which satisfy a logic of said programmable logic array upon completion of precharge period.

25. A data processing system according to claim 24, wherein said programmable logic array has its output lines connected together so as to achieve a logical sum function for the array outputs.

FIG. 1

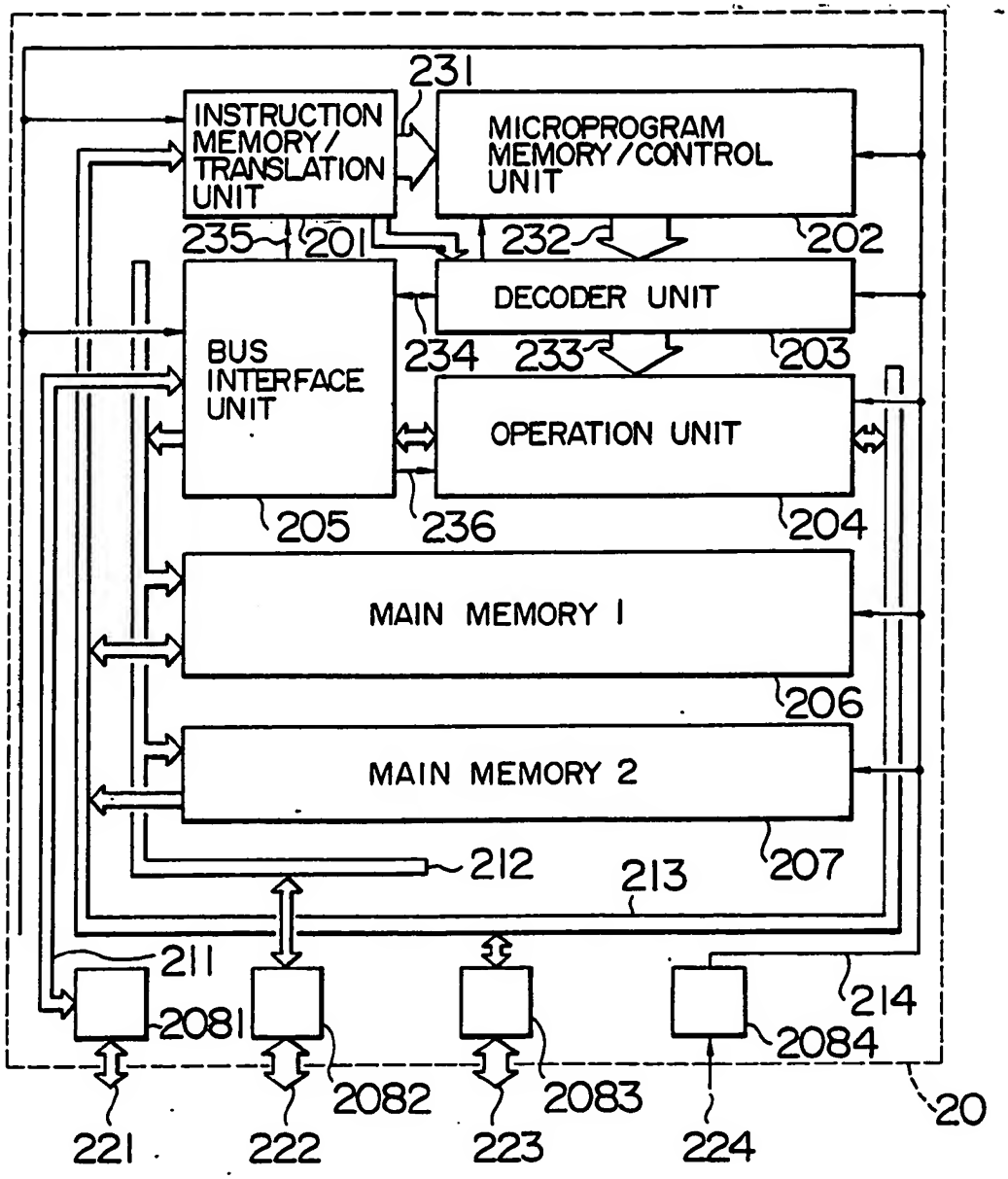

231
INSTRUCTION MEMORY/ TRANSLATION UNIT
MICROPROGRAM MEMORY/CONTROL UNIT
235
201
232
202
DECODER UNIT
BUS INTERFACE UNIT
234
233
203
OPERATION UNIT
205
236
204
MAIN MEMORY 1
206
MAIN MEMORY 2
212
213
207
211
214
2081
2084
2082
2083
20
221
222
223
224

FIG. 2

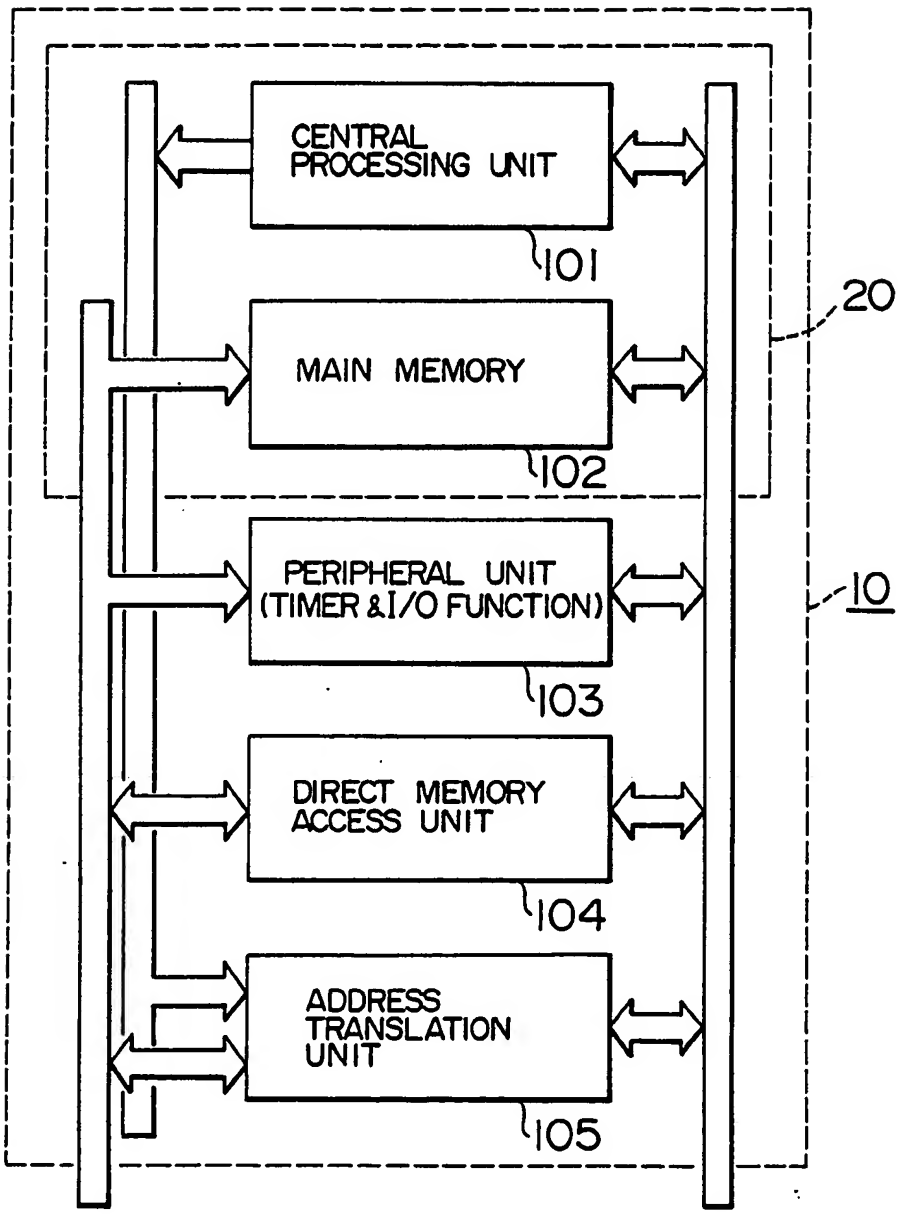
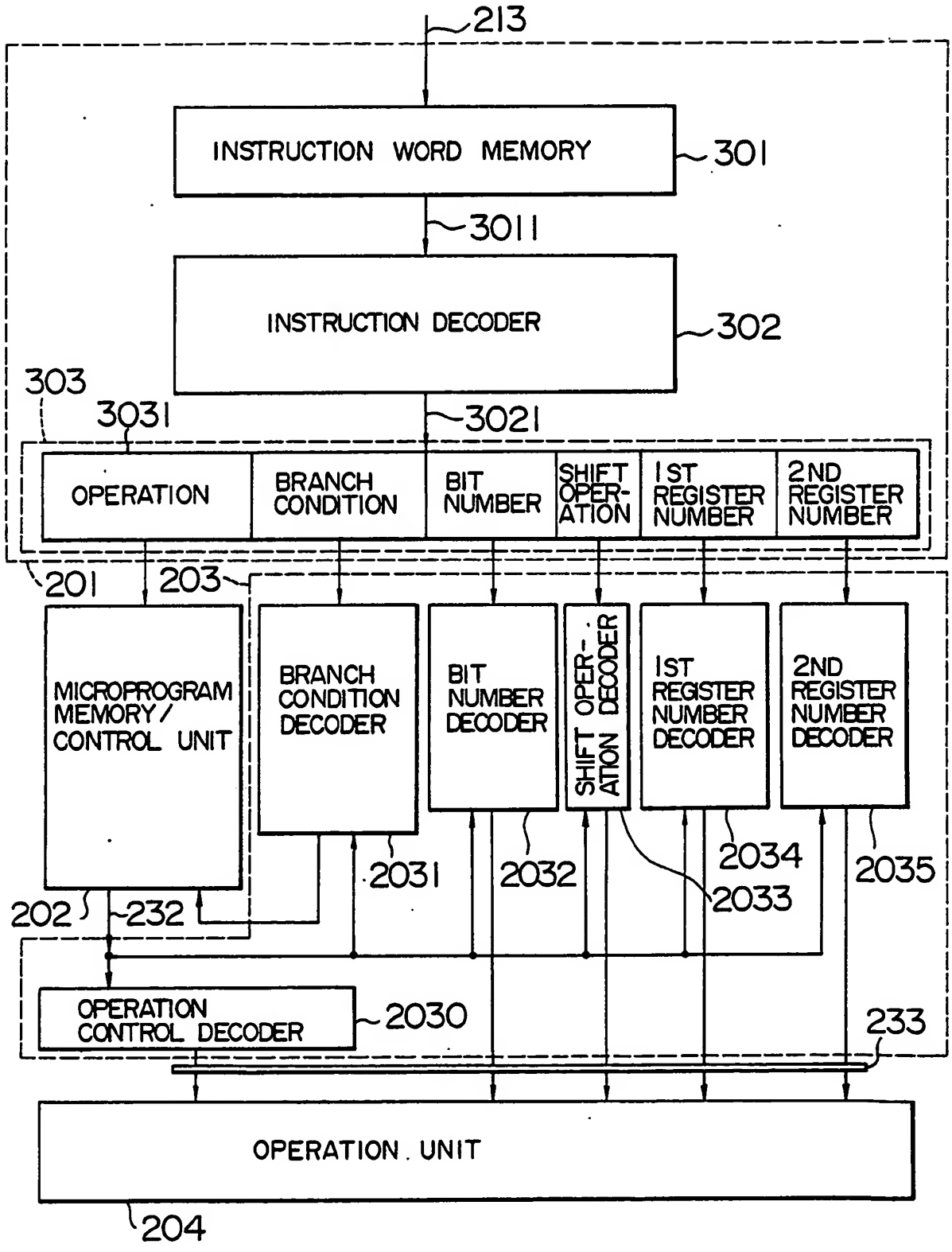

CENTRAL PROCESSING UNIT
101
MAIN MEMORY
102
20
PERIPHERAL UNIT (TIMER &I/O FUNCTION)
103
10
DIRECT MEMORY ACCESS UNIT
104
ADDRESS TRANSLATION UNIT
105

FIG. 3


213
INSTRUCTION WORD MEMORY
301
3011
INSTRUCTION DECODER
302
303
3031
3021
OPERATION
BRANCH CONDITION
BIT NUMBER
SHIFT OPER-ATION
1ST REGISTER NUMBER
2ND REGISTER NUMBER
201
203
MICROPROGRAM MEMORY/ CONTROL UNIT
BRANCH CONDITION DECODER
BIT NUMBER DECODER
SHIFT OPER-ATION DECODER
1ST REGISTER NUMBER DECODER
2ND REGISTER NUMBER DECODER
2031
2032
2033
2034
2035
202
232
OPERATION CONTROL DECODER
2030
233
OPERATION UNIT
204

# FIG. 4

(a) OPERATIONAL INSTRUCTIONS

| OPERATION CODE | | | 1ST OPERAND CODE | 2ND OPERAND CODE | |
|---|---|---|---|---|---|
| OPERATION CLASS CODE | OPERATION DATA SIZE | OPERATION TYPE | 1ST REGISTER NUMBER | ADDRESSING CODE | 2ND REGISTER NUMBER |

(b) BRANCHING INSTRUCTIONS

| OPERATION CODE | | 1ST OPERAND CODE | |
|---|---|---|---|
| OPERATION CLASS CODE | BRANCH CONDITION | ADDRESSING CODE | 1ST REGISTER NUMBER |

(c) BIT MANIPULATING INSTRUCTIONS

| OPERATION CODE | | 1ST OPERAND CODE | 2ND OPERAND CODE | |
|---|---|---|---|---|
| OPERATION CLASS CODE | BIT MANIPULATION TYPE | BIT NUMBER | ADDRESSING CODE | 2ND REGISTER NUMBER |

FIG. 5

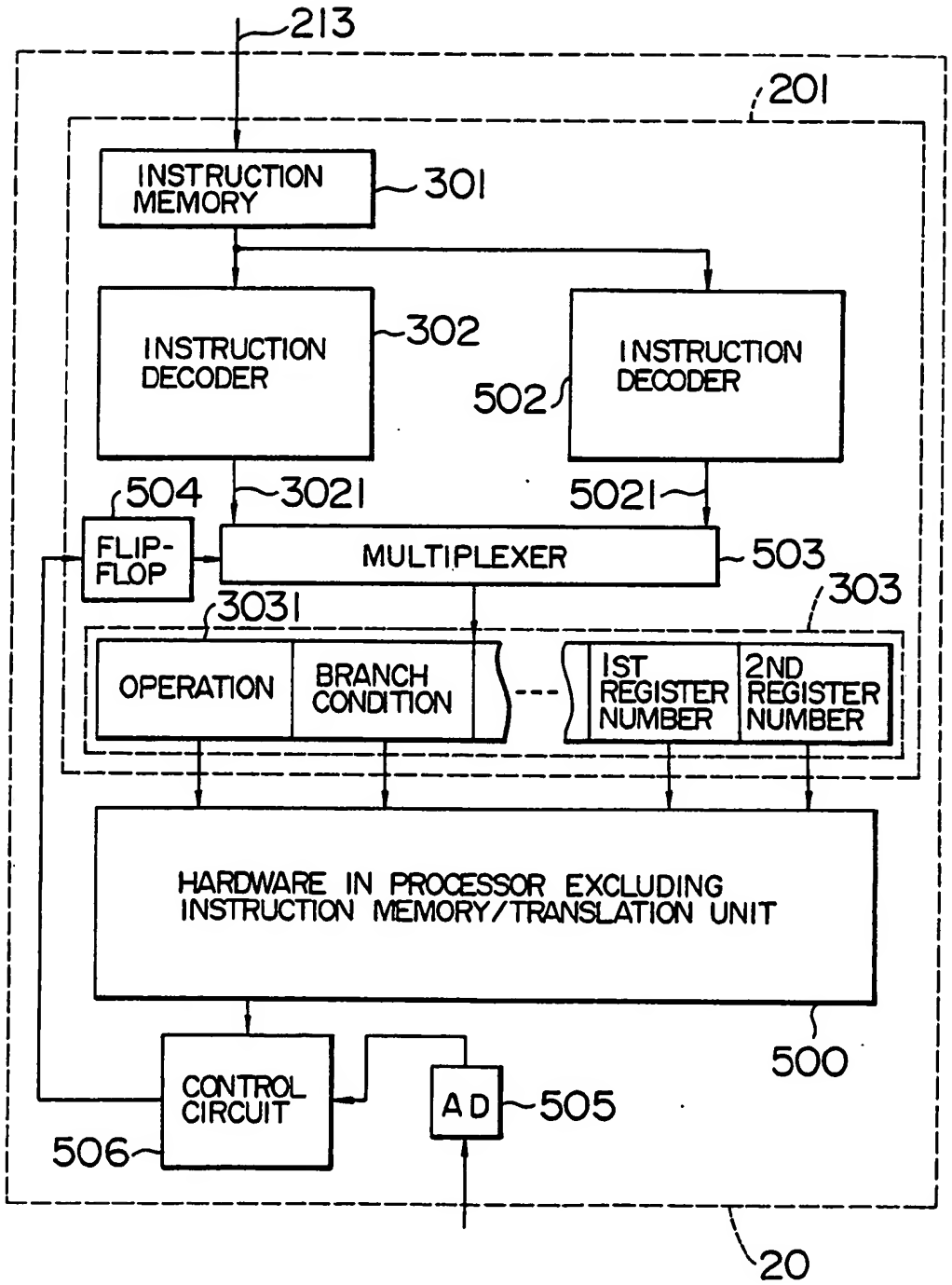

213
201
INSTRUCTION MEMORY
301
INSTRUCTION DECODER
302
502
INSTRUCTION DECODER
504
3021
5021
FLIP-FLOP
MULTIPLEXER
503
303
3031
OPERATION
BRANCH CONDITION
1ST REGISTER NUMBER
2ND REGISTER NUMBER
HARDWARE IN PROCESSOR EXCLUDING INSTRUCTION MEMORY/TRANSLATION UNIT
500
CONTROL CIRCUIT
AD
505
506
20

FIG. 6

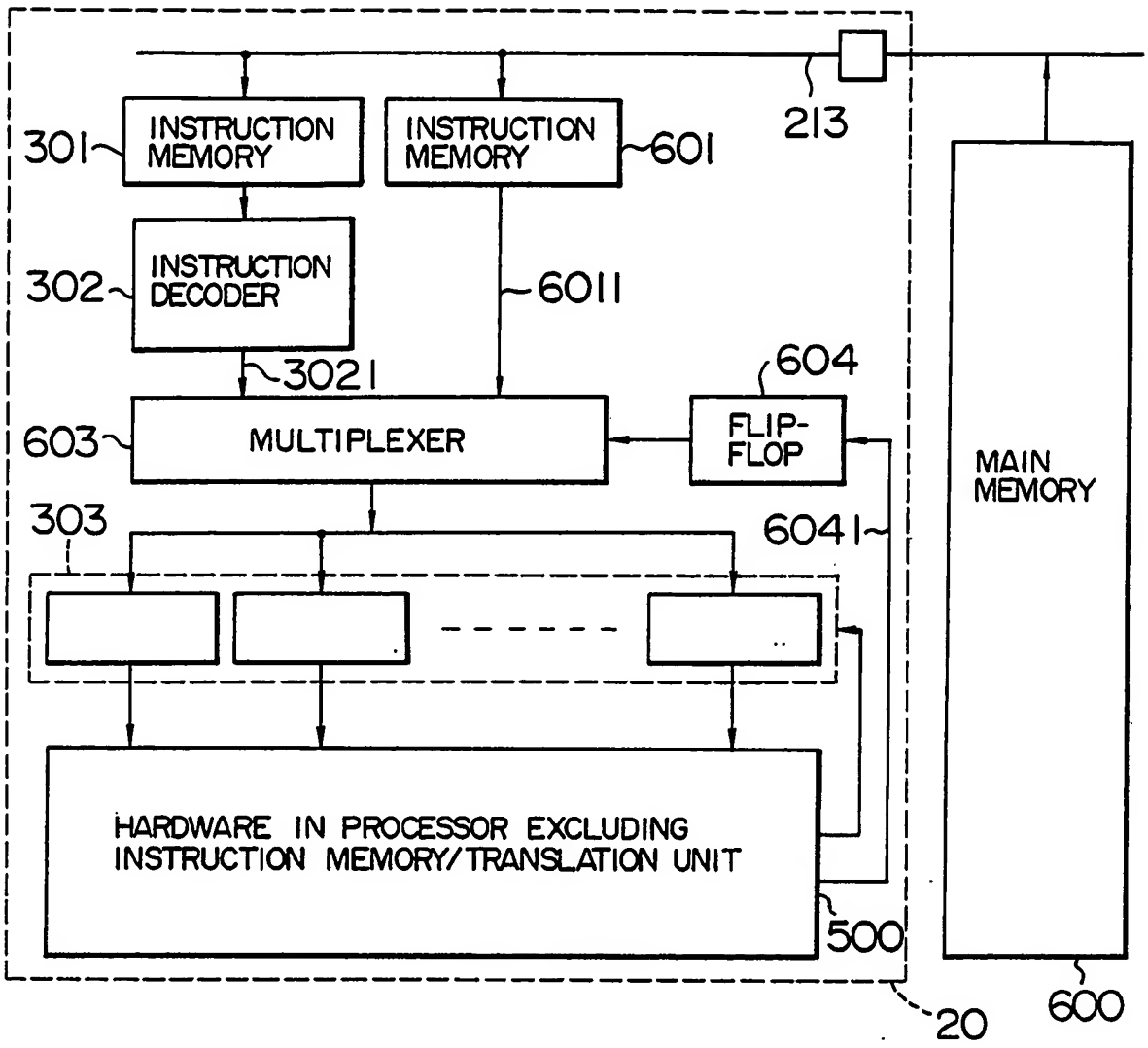

213
301
INSTRUCTION MEMORY
INSTRUCTION MEMORY
601
302
INSTRUCTION DECODER
6011
3021
604
603
MULTIPLEXER
FLIP-FLOP
MAIN MEMORY
303
6041
HARDWARE IN PROCESSOR EXCLUDING INSTRUCTION MEMORY/TRANSLATION UNIT
500
20
600

FIG. 7

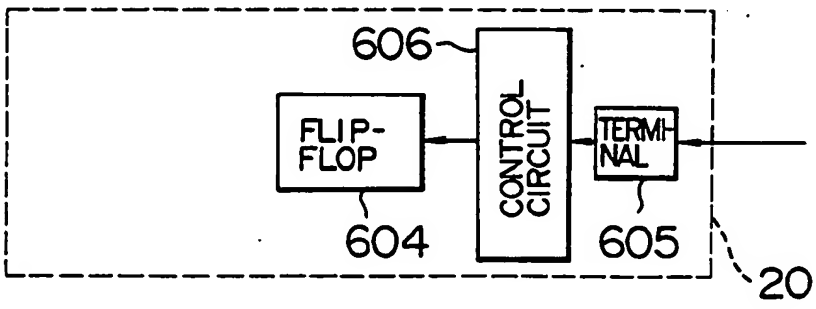

606
FLIP-FLOP
CONTROL CIRCUIT
TERMINAL
604
605
20

FIG. 8

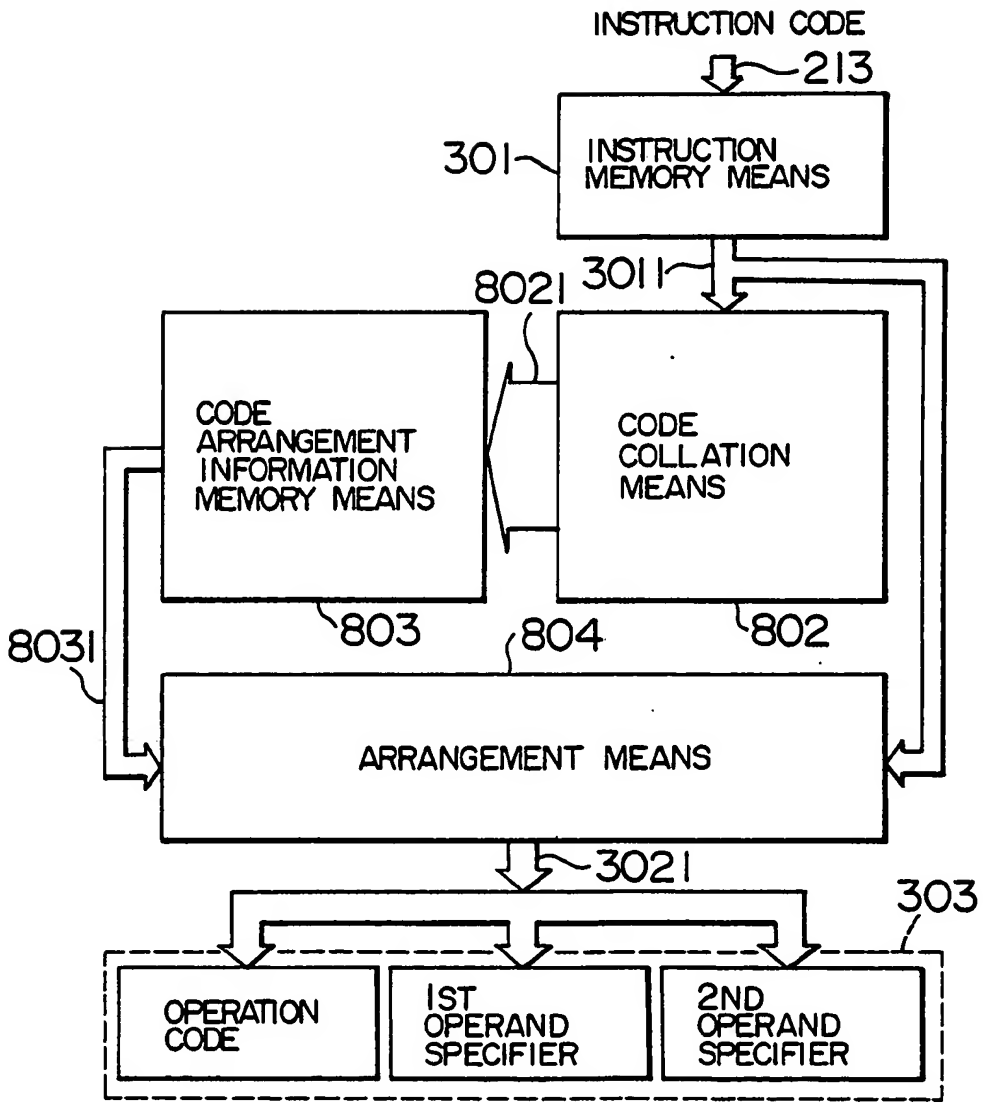

INSTRUCTION CODE
213
301
INSTRUCTION MEMORY MEANS
8021
3011
CODE ARRANGEMENT INFORMATION MEMORY MEANS
CODE COLLATION MEANS
803
804
802
8031
ARRANGEMENT MEANS
3021
303
OPERATION CODE
1ST OPERAND SPECIFIER
2ND OPERAND SPECIFIER

FIG. 9

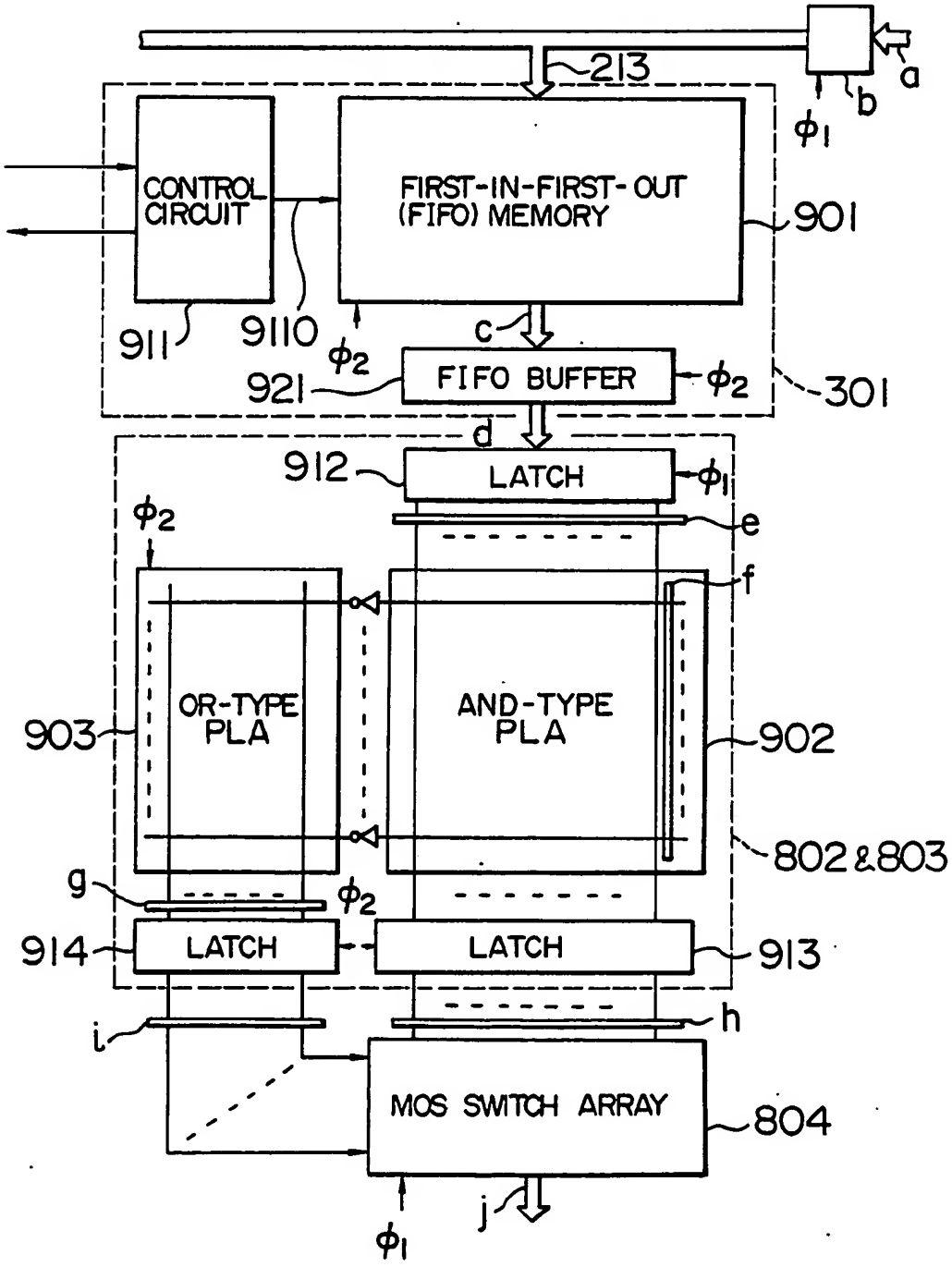

213
a
b
$\phi_1$
CONTROL CIRCUIT
FIRST-IN-FIRST-OUT (FIFO) MEMORY
901
911
9110
$\phi_2$
c
921
FIFO BUFFER
$\phi_2$
301
d
912
LATCH
$\phi_1$
$\phi_2$
e
f
OR-TYPE PLA
AND-TYPE PLA
903
902
802 & 803
g
$\phi_2$
914
LATCH
LATCH
913
h
i
MOS SWITCH ARRAY
804
j
$\phi_1$

FIG. 10

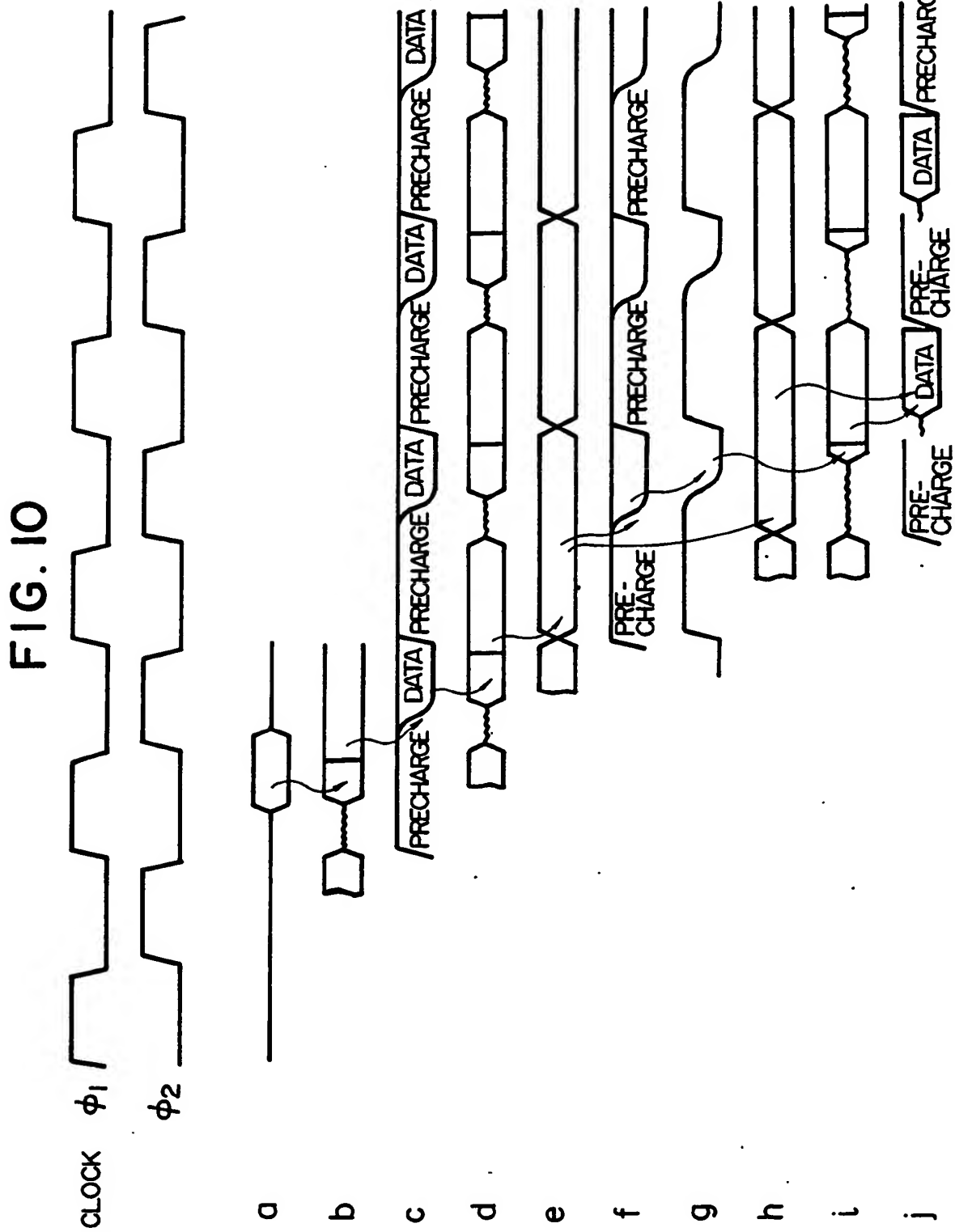

BASIC CLOCK
$\phi_1$
$\phi_2$
a
b
c
d
e
f
g
h
i
j
PRECHARGE
DATA
PRE-CHARGE

FIG. 11

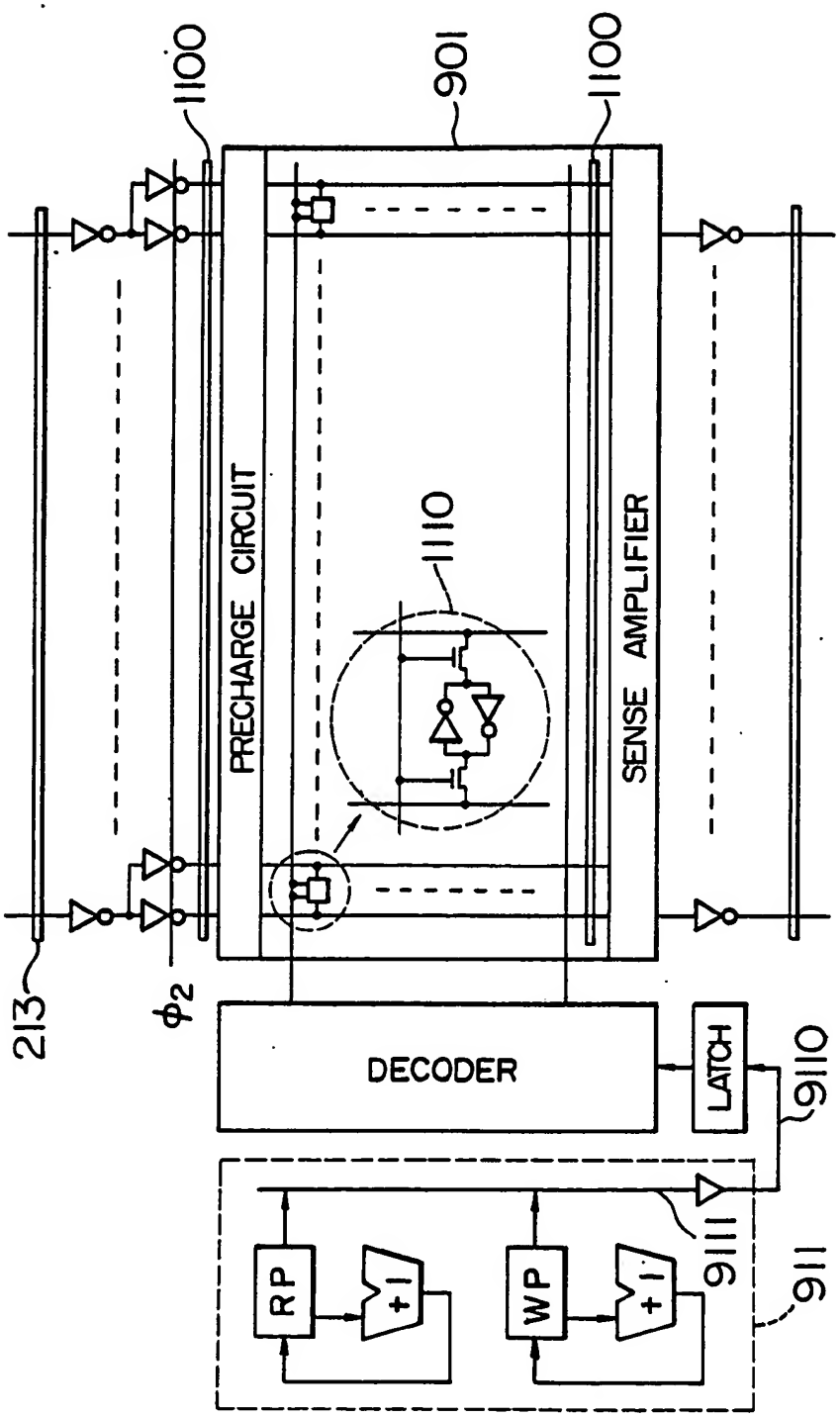

1100
901
1100
PRECHARGE CIRCUIT
1110
SENSE AMPLIFIER
213
$\phi_2$
DECODER
LATCH
9110
RP
+1
WP
+1
9111
911

FIG. 12

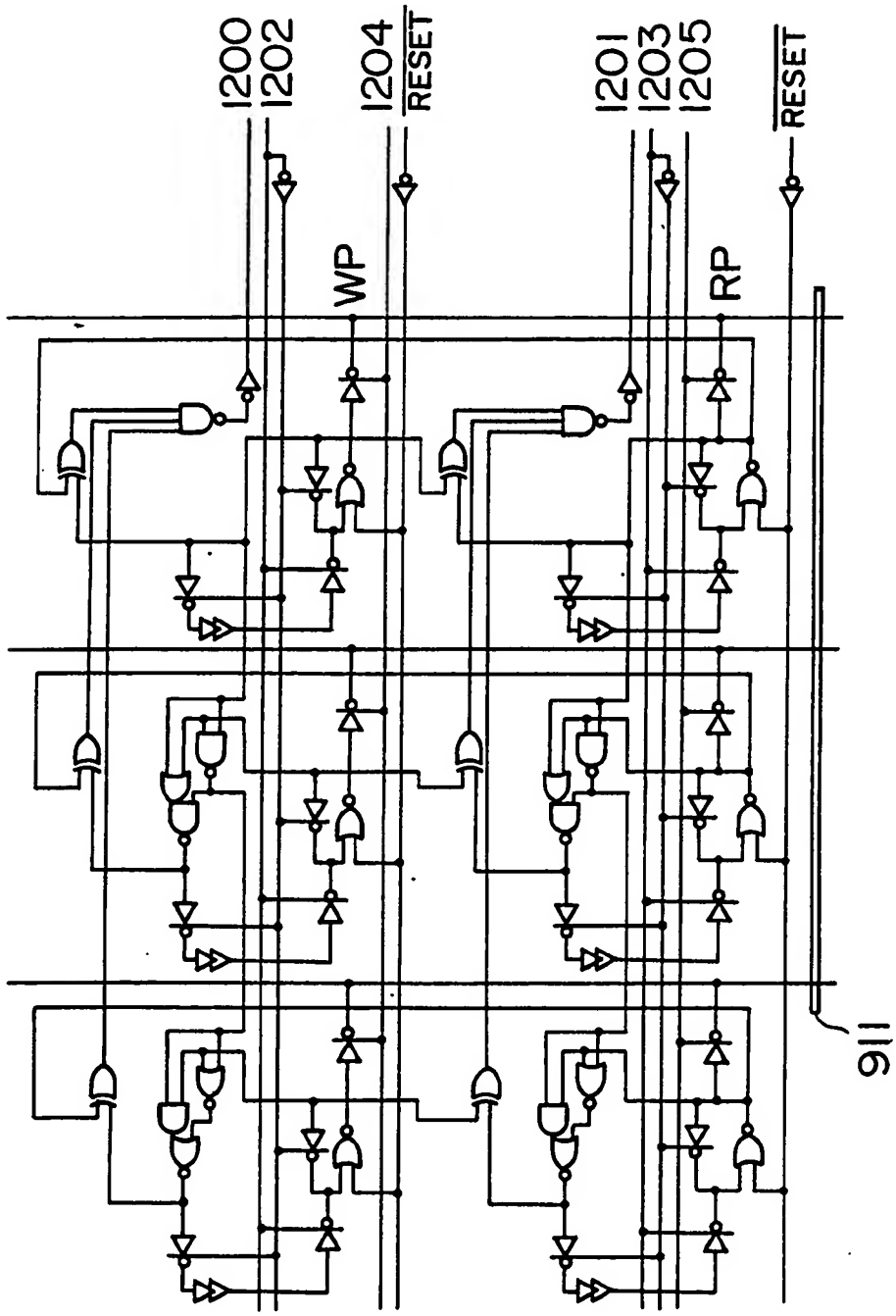


FIG. 13

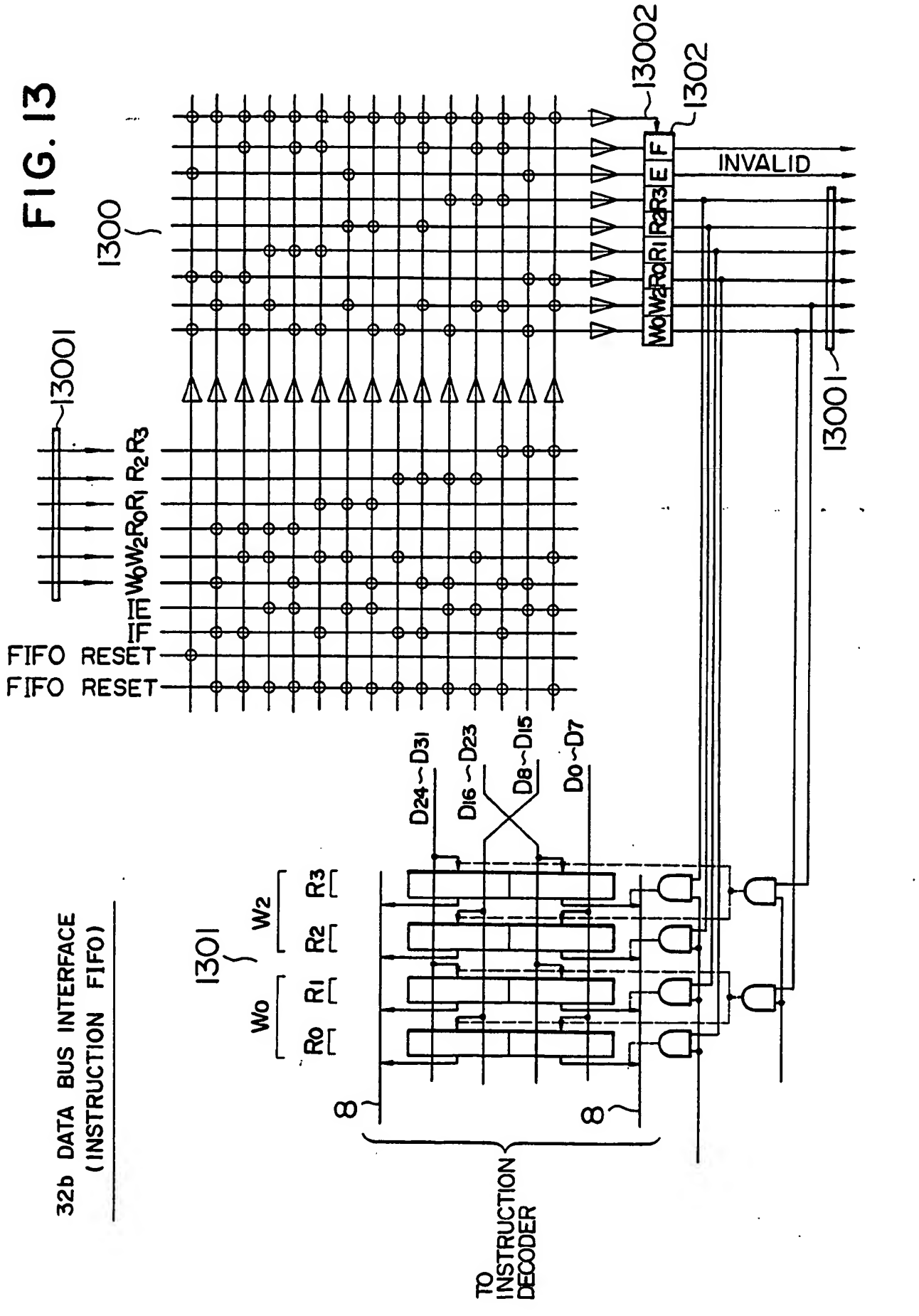

1300
1302
13002
INVALID
13001
13001
W0 W2 R0 R1 R2 R3
FIFO RESET
FIFO RESET
D24~D31
D16~D23
D8~D15
D0~D7
1301
W0
W2
R0
R1
R2
R3
8
8
TO INSTRUCTION DECODER
32b DATA BUS INTERFACE (INSTRUCTION FIFO)

FIG. 14

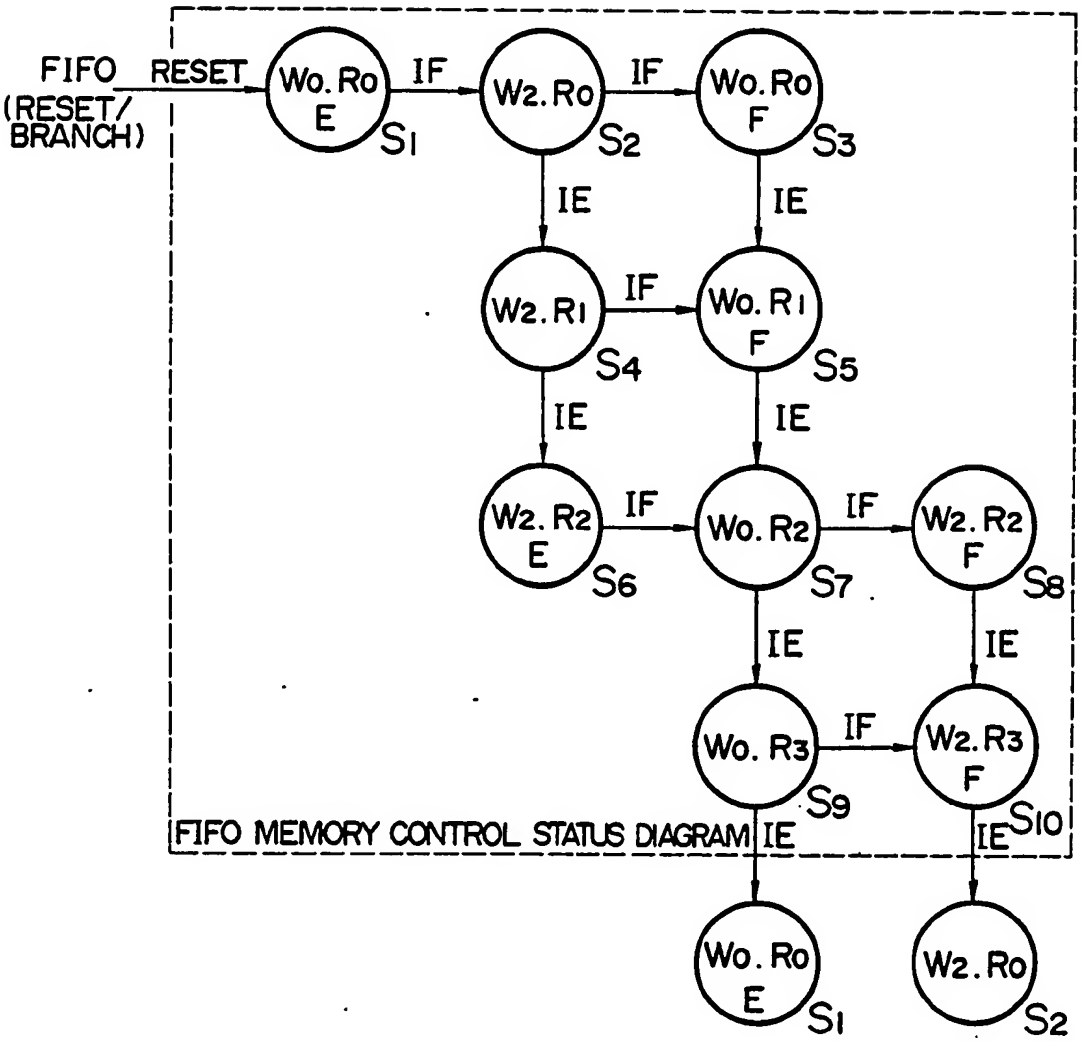

FIFO
(RESET/
BRANCH)
RESET
Wo. Ro
E
S1
IF
W2. Ro
S2
IF
Wo. Ro
F
S3
IE
IE
W2. R1
S4
IF
Wo. R1
F
S5
IE
IE
W2. R2
E
S6
IF
Wo. R2
S7
IF
W2. R2
F
S8
IE
IE
Wo. R3
S9
IF
W2. R3
F
S10
IE
IE
Wo. Ro
E
S1
W2. Ro
S2
FIFO MEMORY CONTROL STATUS DIAGRAM

FIG. 15

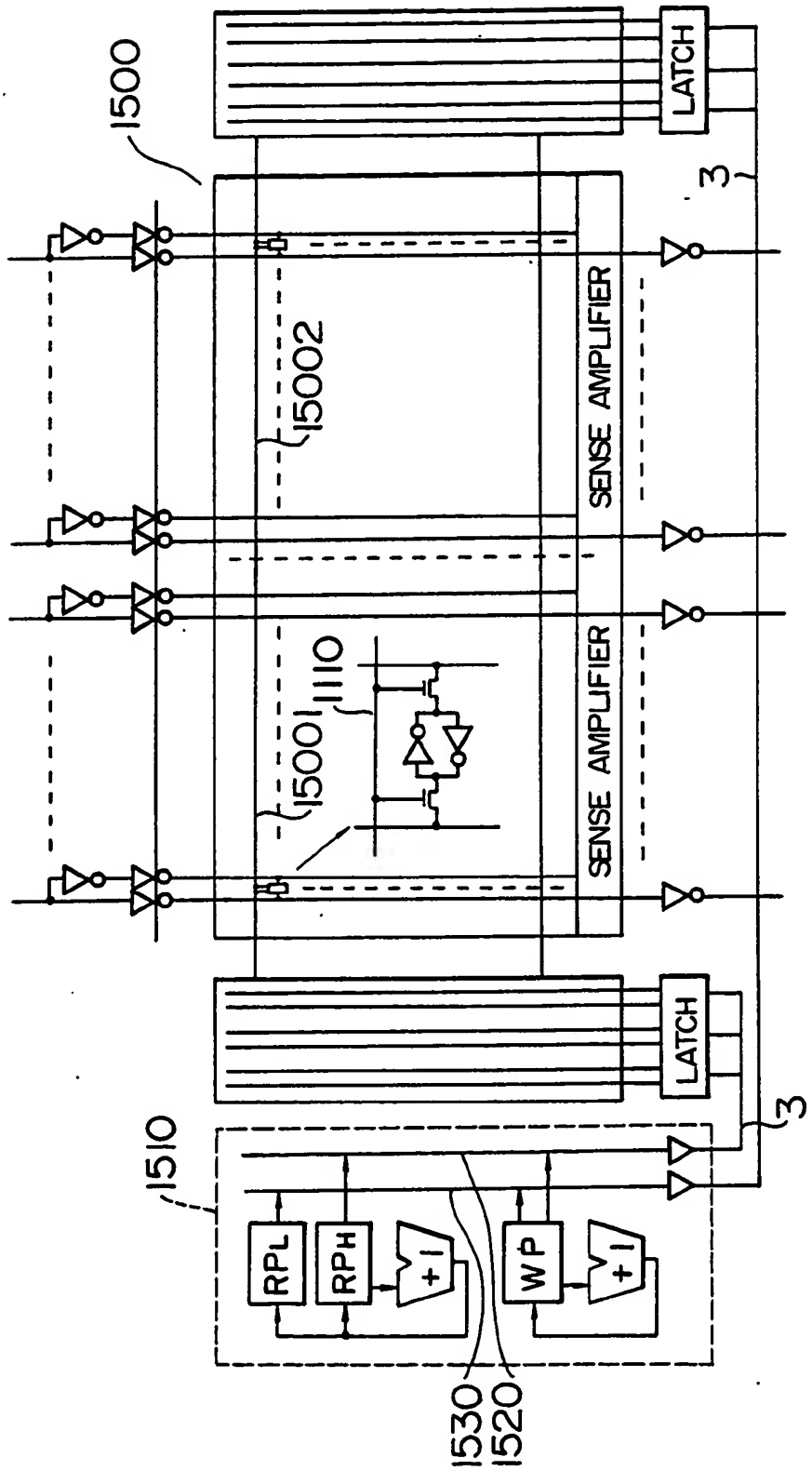

1500
LATCH
3
15002
SENSE AMPLIFIER
1110
15001
SENSE AMPLIFIER
LATCH
3
1510
RPL
RPH
+1
WP
+1
1530
1520

FIG. 16

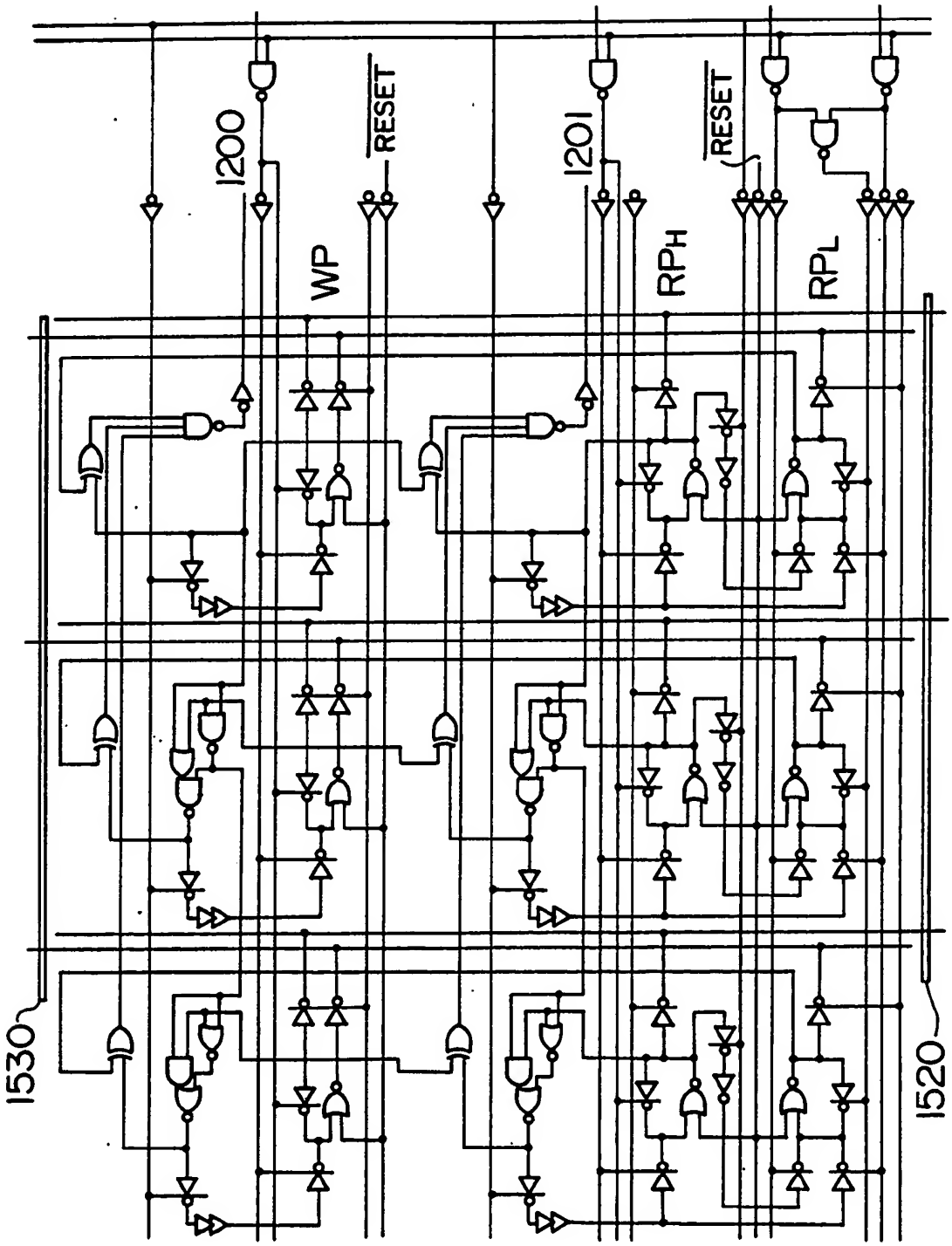


FIG. 17

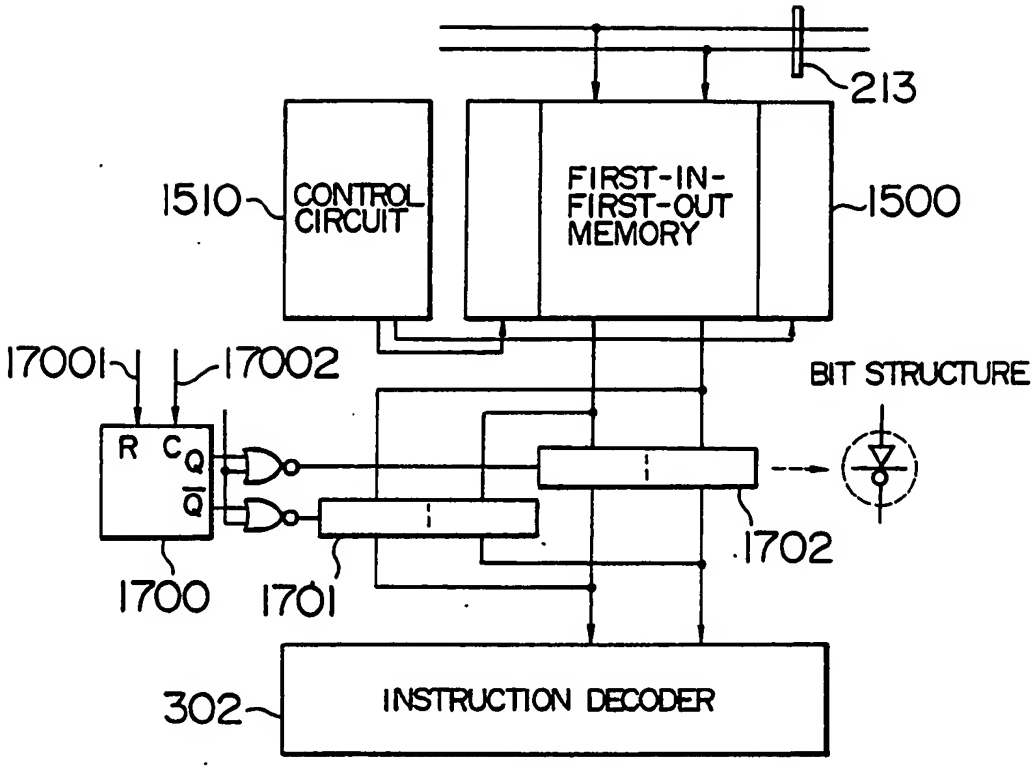



FIG. 18

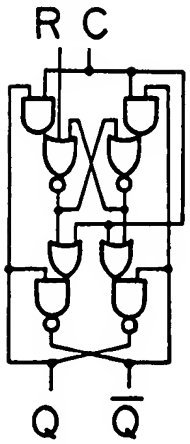



FIG. 19

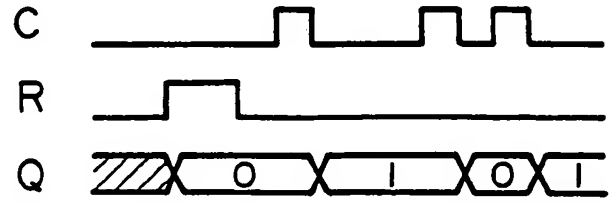


FIG. 20

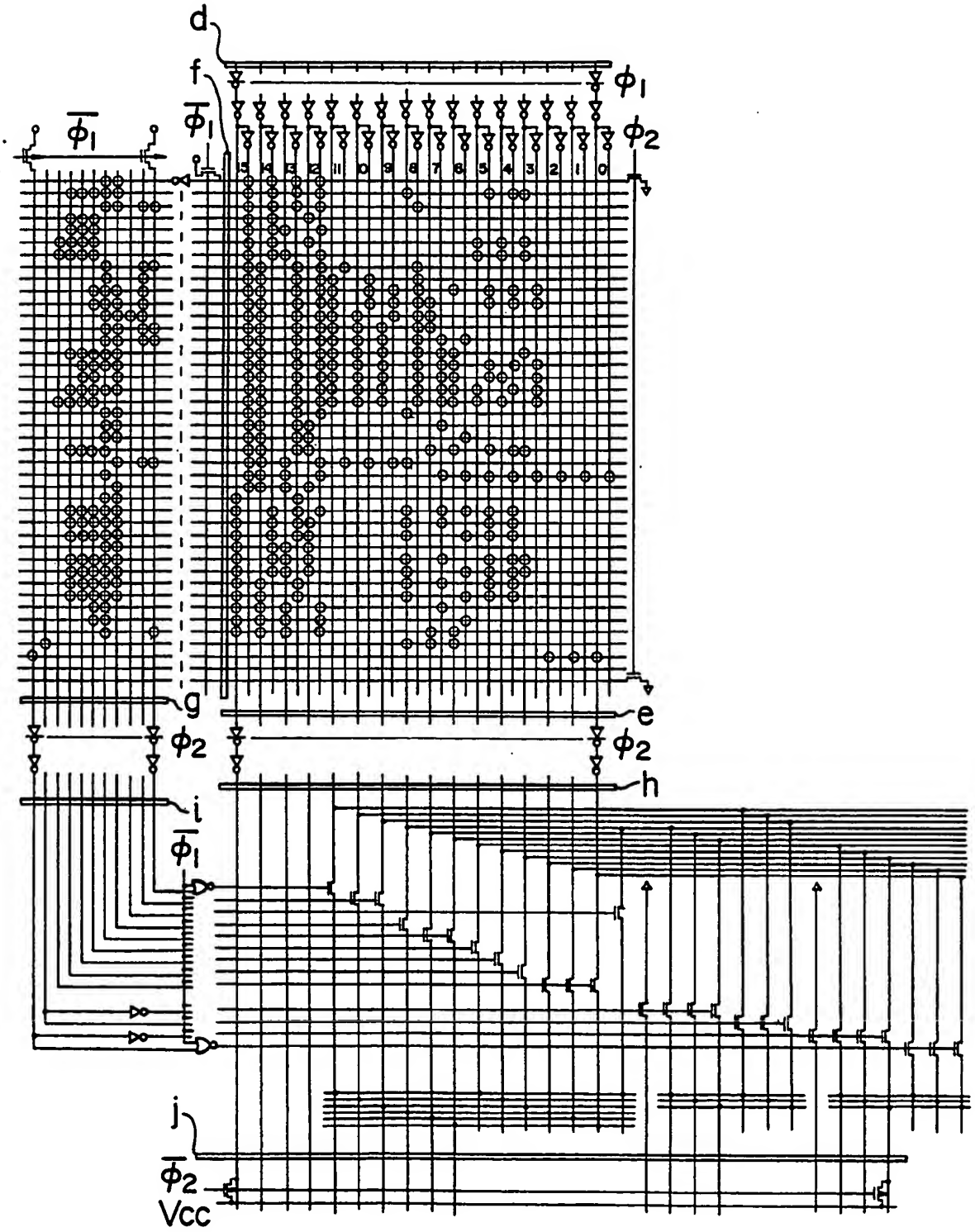

d
f
$\overline{\phi_1}$
$\phi_1$
$\overline{\phi_1}$
$\phi_2$
15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0
g
e
$\phi_2$
$\phi_2$
h
i
$\overline{\phi_1}$
j
$\overline{\phi_2}$
Vcc

FIG. 21

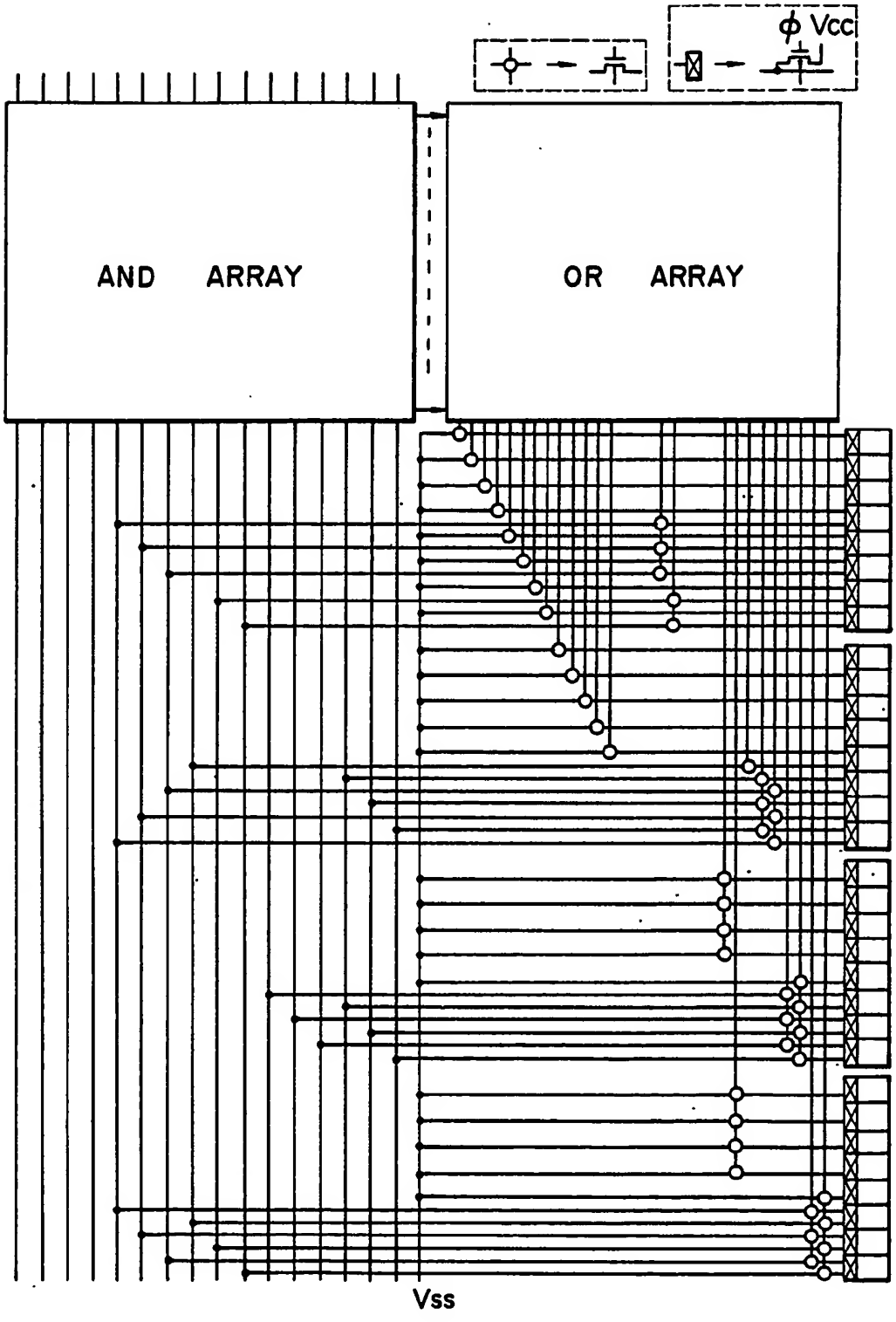


FIG. 22

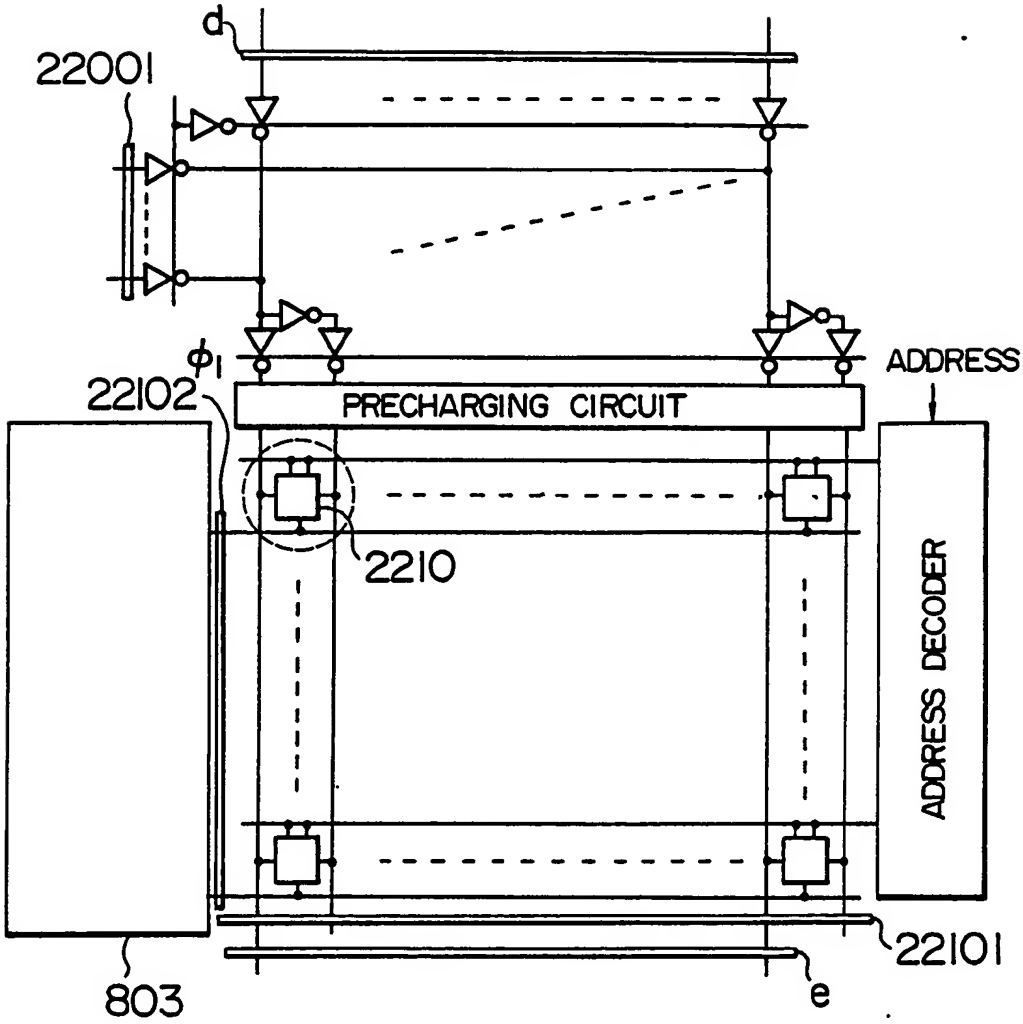

d
22001
$\phi_1$
22102
PRECHARGING CIRCUIT
ADDRESS
2210
ADDRESS DECODER
22101
803
e

FIG. 23

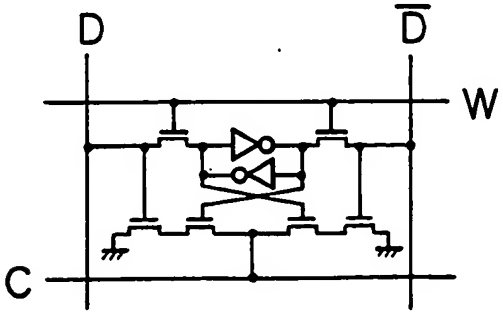

D
$\overline{D}$
W
C

FIG. 24

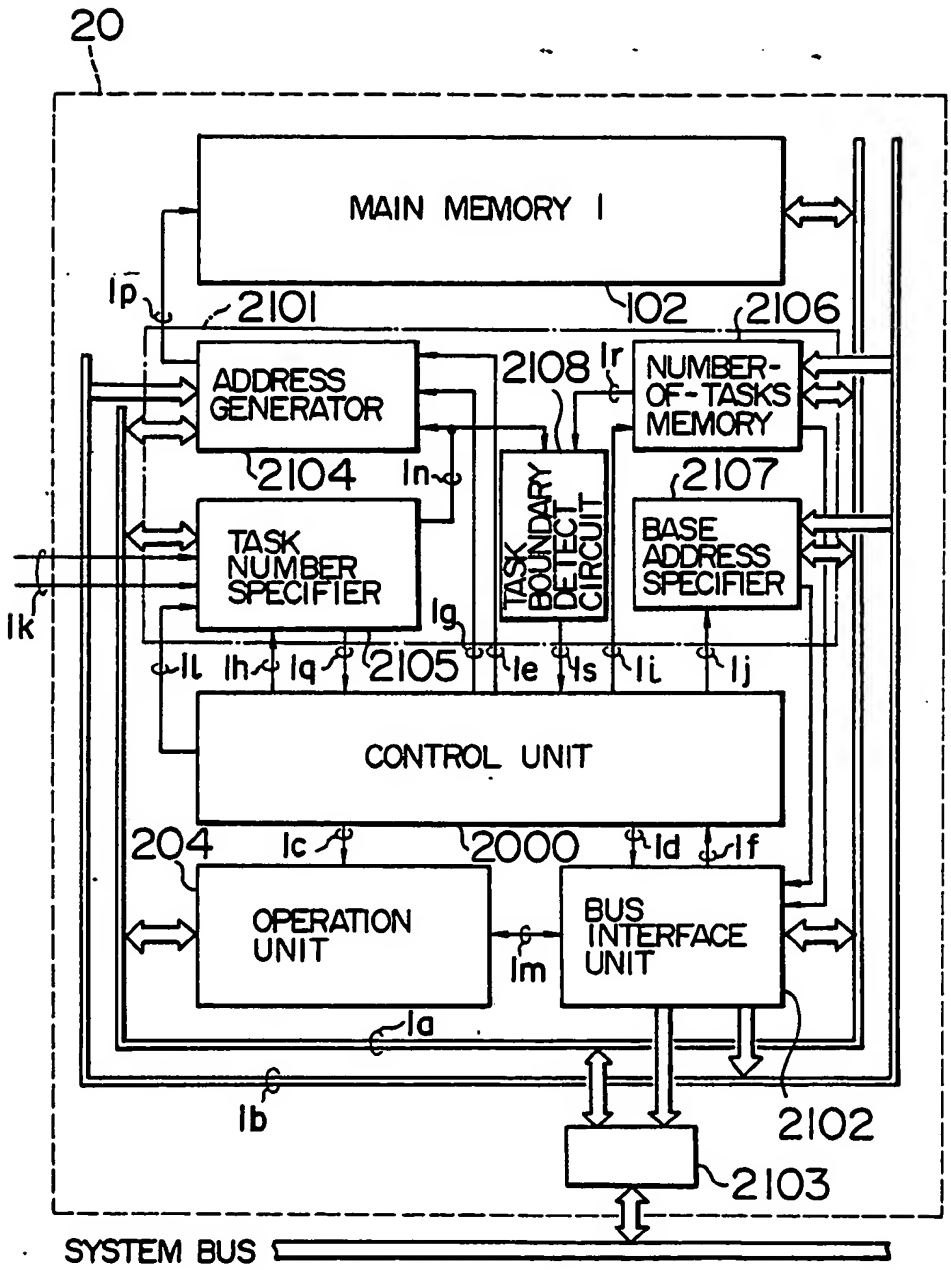

20
MAIN MEMORY 1
102
2101
2106
ADDRESS GENERATOR
2104
2108
NUMBER-OF-TASKS MEMORY
2107
TASK NUMBER SPECIFIER
TASK BOUNDARY DETECT CIRCUIT
BASE ADDRESS SPECIFIER
2105
CONTROL UNIT
2000
204
OPERATION UNIT
BUS INTERFACE UNIT
2102
2103
SYSTEM BUS
Ia
Ib
Ic
Id
Ie
If
Ig
Ih
Ii
Ij
Ik
Il
Im
In
Ip
Iq
Ir
Is

FIG. 25

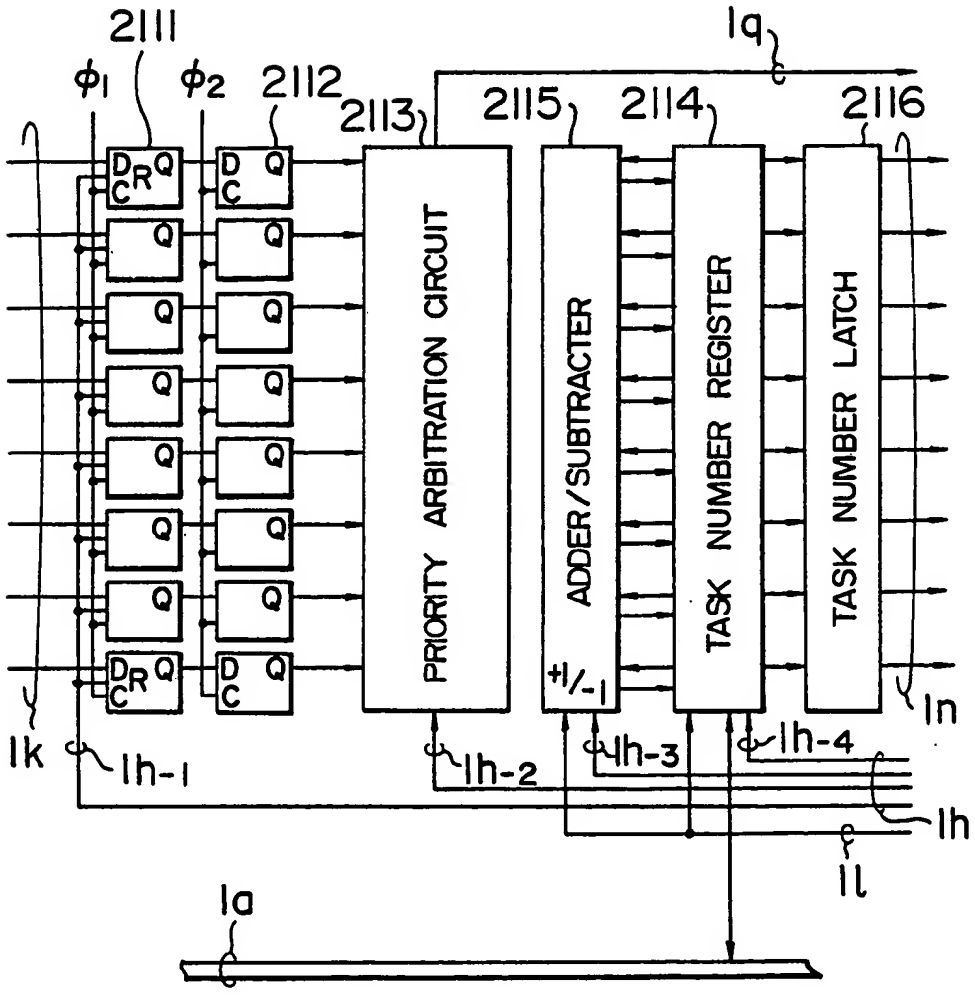
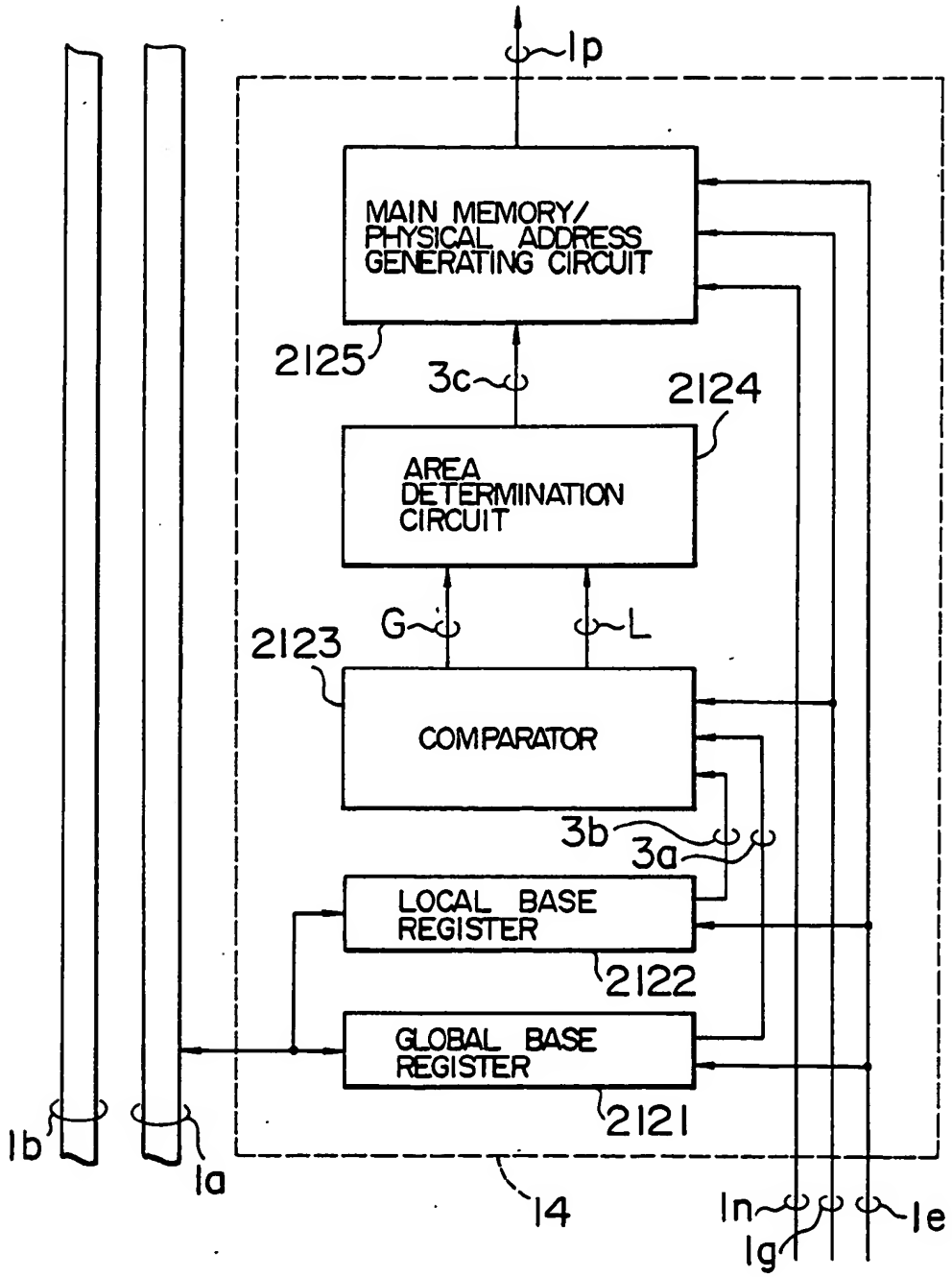
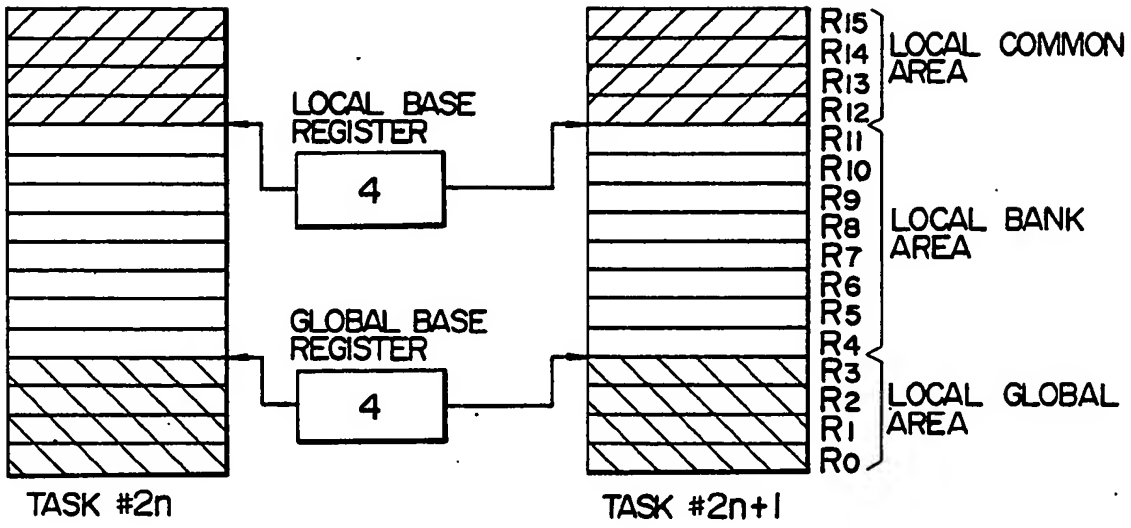

2111
φ1
φ2
2112
2113
2115
2114
2116
1q
PRIORITY ARBITRATION CIRCUIT
ADDER/SUBTRACTER
+1/-1
TASK NUMBER REGISTER
TASK NUMBER LATCH
1k
1h-1
1h-2
1h-3
1h-4
1h
1l
1n
1a

FIG. 26


1p
MAIN MEMORY/
PHYSICAL ADDRESS
GENERATING CIRCUIT
2125
3c
2124
AREA
DETERMINATION
CIRCUIT
G
L
2123
COMPARATOR
3b
3a
LOCAL BASE
REGISTER
2122
GLOBAL BASE
REGISTER
2121
1b
1a
14
1n
1e
1g

# FIG. 27

(a) SWITCHING FROM TASK #2n TO TASK #2n+1 (n=0,1,2,···)




(b) SWITCHING FROM TASK #2n+1 TO TASK #2n+2 (n=0,1,2,···)

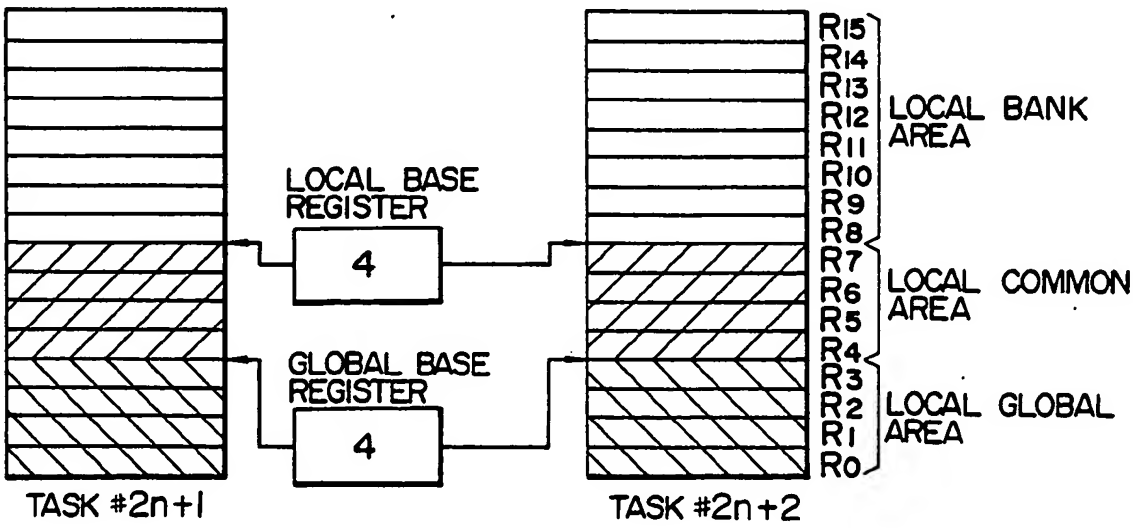


FIG. 28

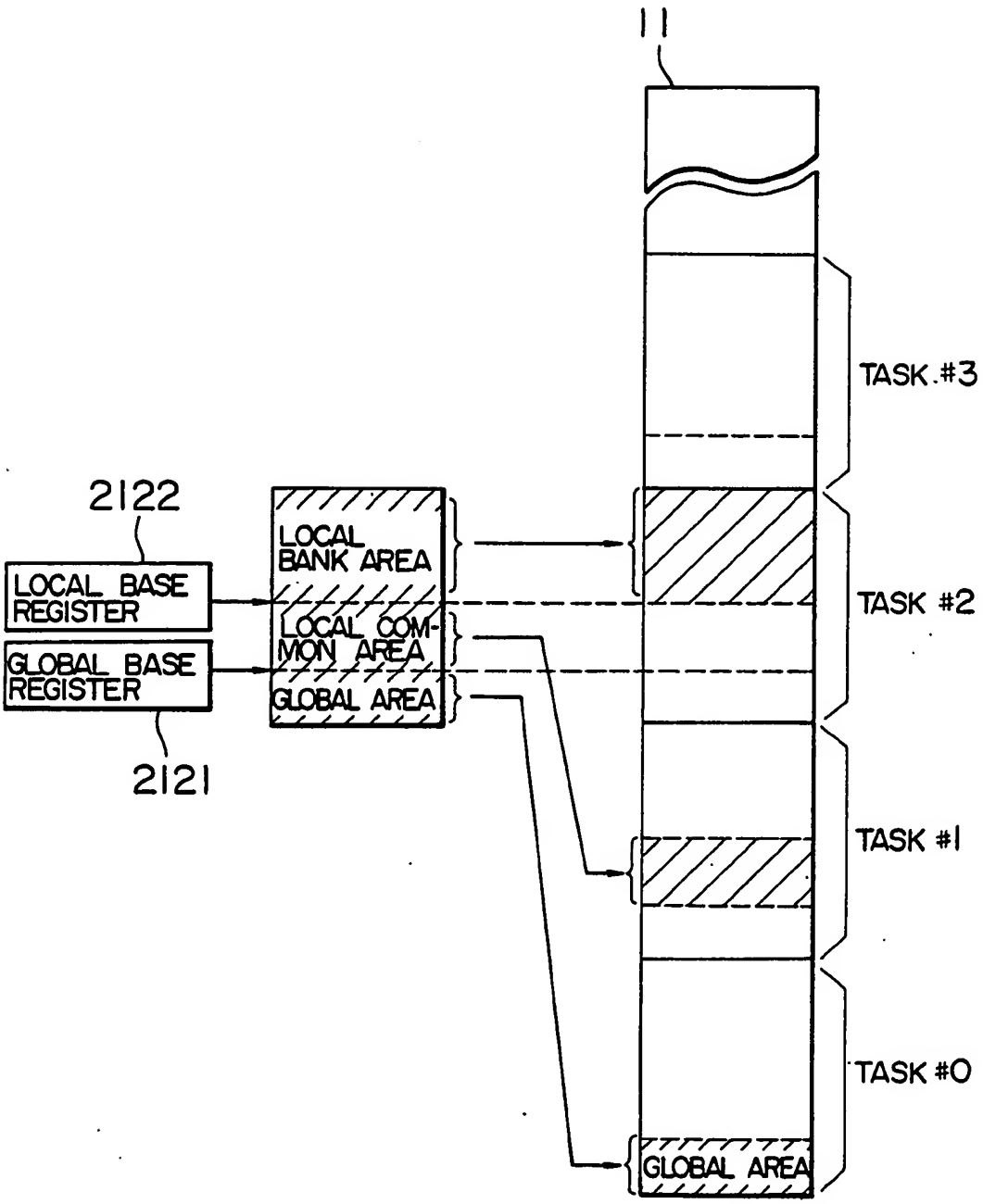

11
TASK #3
TASK #2
TASK #1
TASK #0
2122
LOCAL BASE REGISTER
GLOBAL BASE REGISTER
2121
LOCAL BANK AREA
LOCAL COM-MON AREA
GLOBAL AREA
GLOBAL AREA

FIG. 29

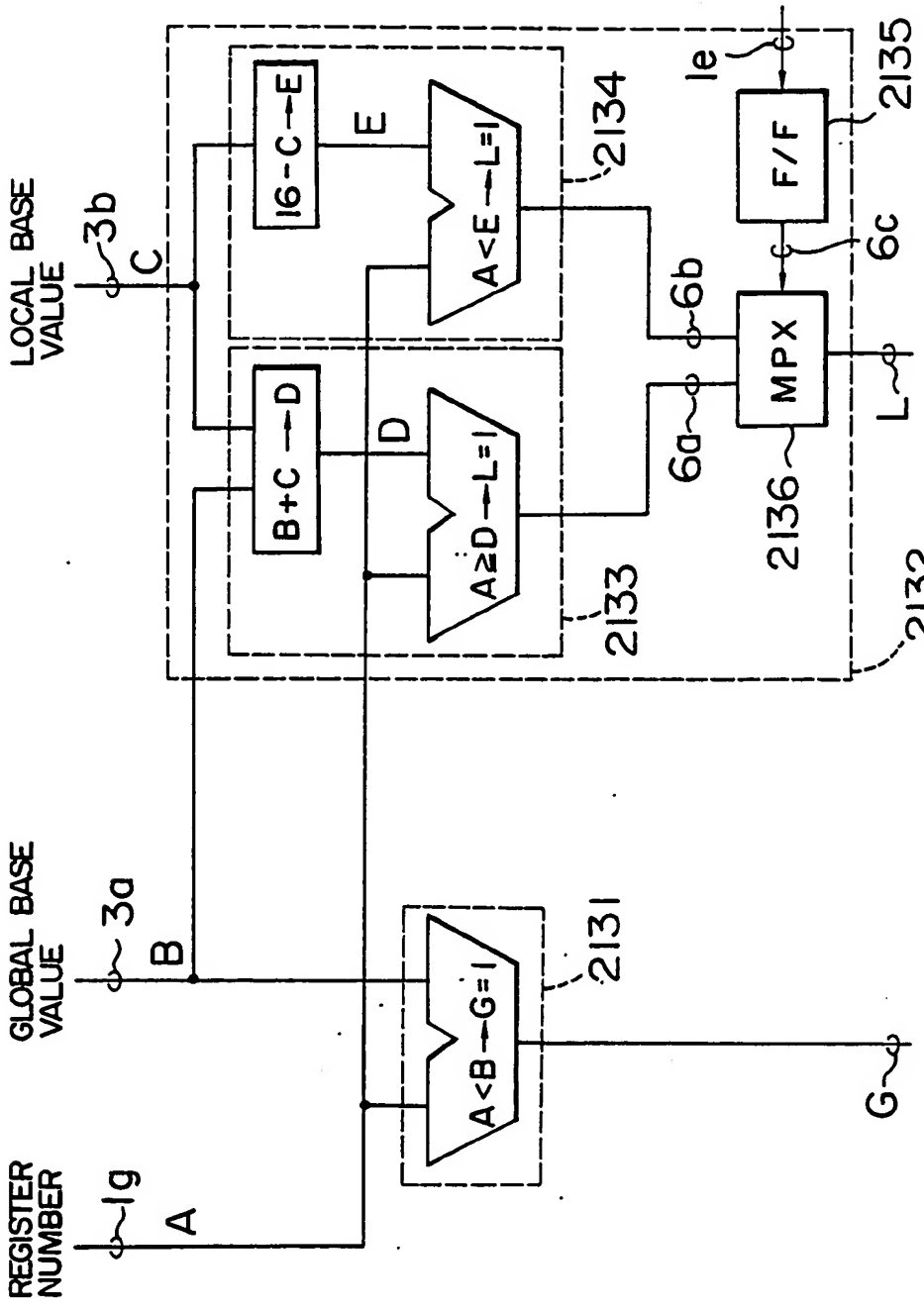
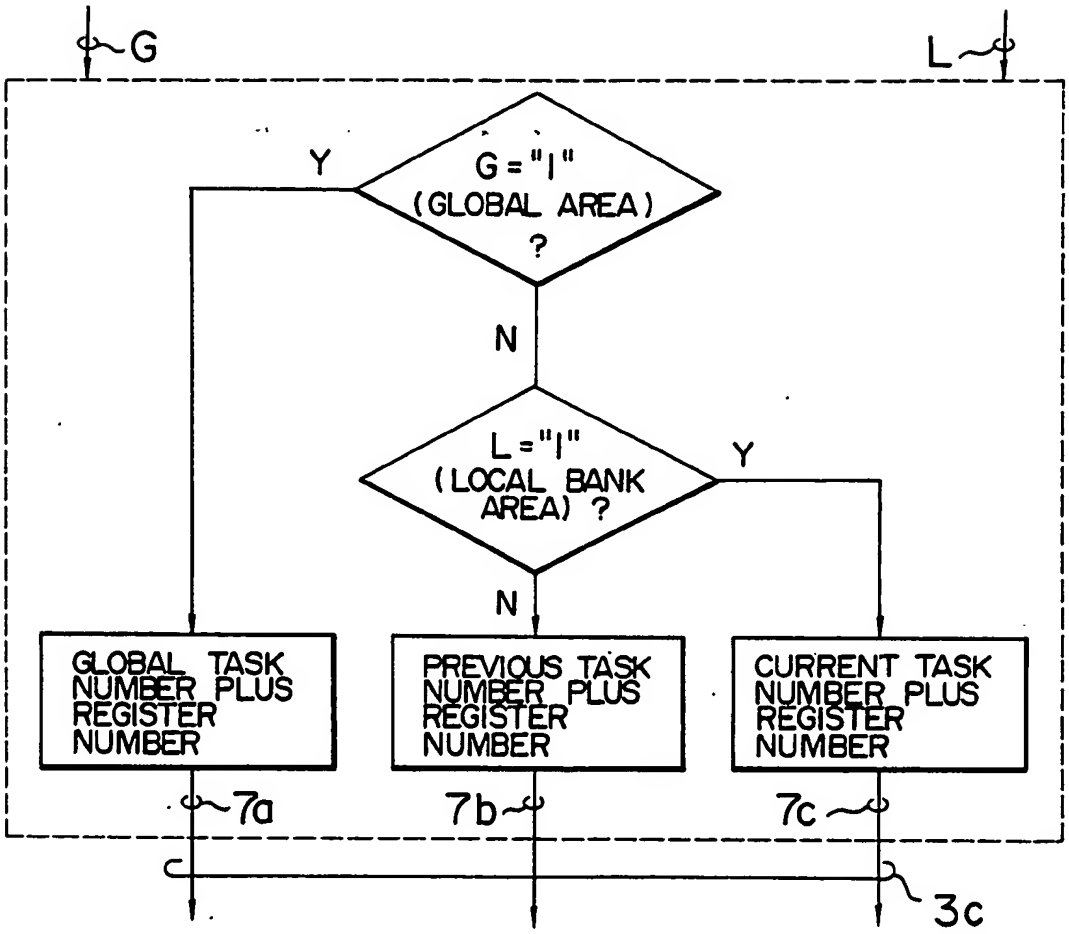
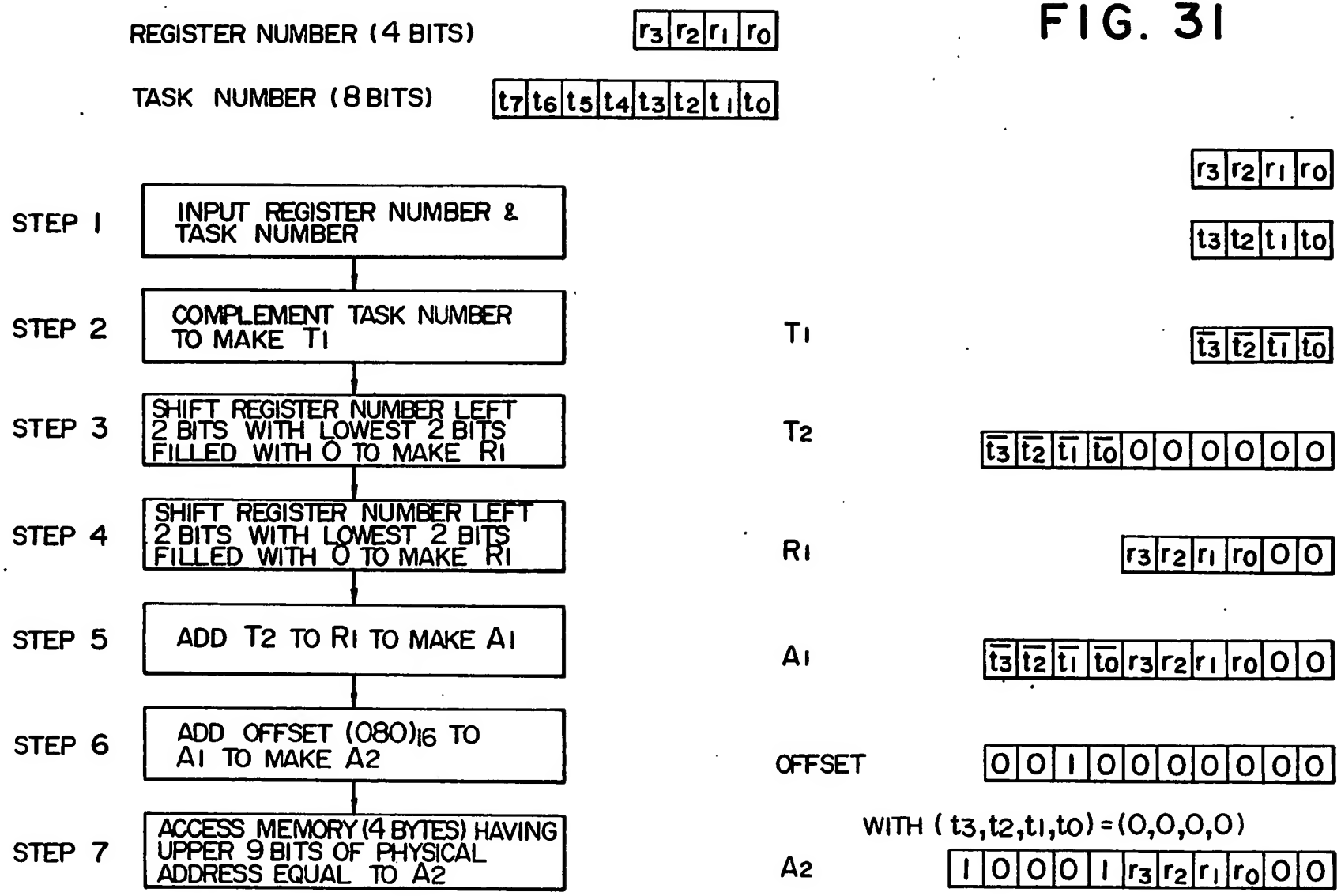

REGISTER NUMBER
1g
A
GLOBAL BASE VALUE
3a
B
LOCAL BASE VALUE
3b
C
A<B→G=1
2131
G
B+C→D
D
A≥D→L=1
2133
16-C→E
E
A<E→L=1
2134
6a
6b
MPX
L
F/F
1e
2135
6c
2136
2132

FIG. 30


G
L
G = "1" (GLOBAL AREA) ?
Y
N
L = "1" (LOCAL BANK AREA) ?
Y
N
GLOBAL TASK NUMBER PLUS REGISTER NUMBER
PREVIOUS TASK NUMBER PLUS REGISTER NUMBER
CURRENT TASK NUMBER PLUS REGISTER NUMBER
7a
7b
7c
3c

# FIG. 31




0 199 173

FIG. 32

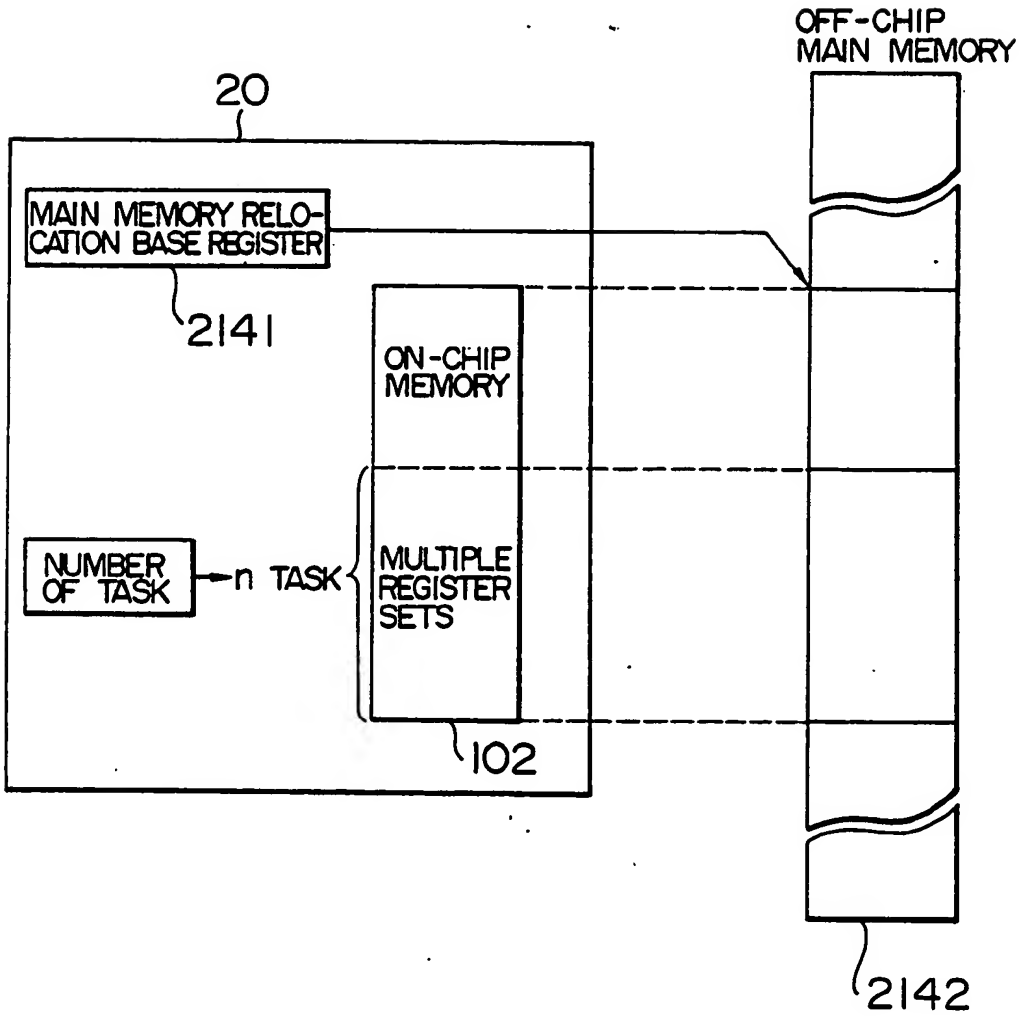

OFF-CHIP
MAIN MEMORY
20
MAIN MEMORY RELO-
CATION BASE REGISTER
2141
ON-CHIP
MEMORY
NUMBER
OF TASK
n TASK
MULTIPLE
REGISTER
SETS
102
2142

FIG. 33

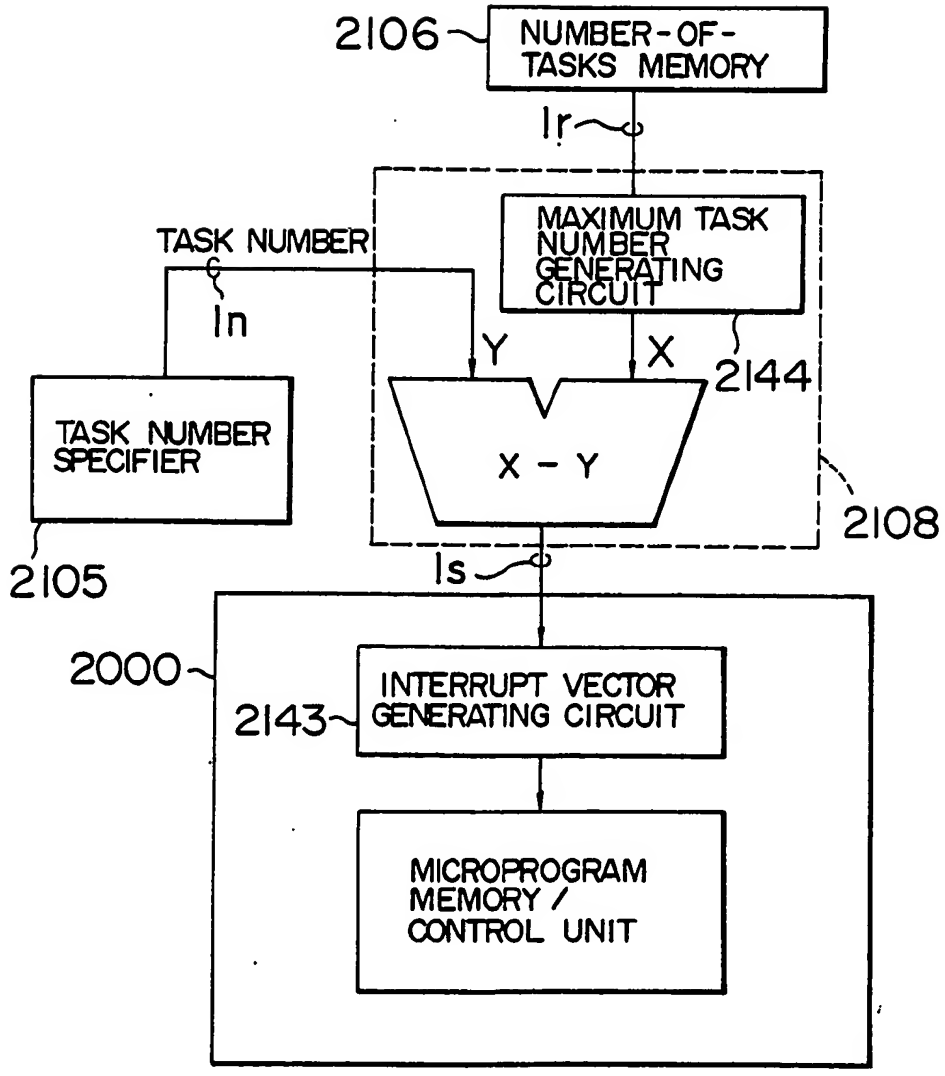

2106
NUMBER-OF-TASKS MEMORY
Ir
MAXIMUM TASK NUMBER GENERATING CIRCUIT
TASK NUMBER
In
Y
X
2144
TASK NUMBER SPECIFIER
X - Y
2108
2105
Is
2000
2143
INTERRUPT VECTOR GENERATING CIRCUIT
MICROPROGRAM MEMORY / CONTROL UNIT

FIG. 34

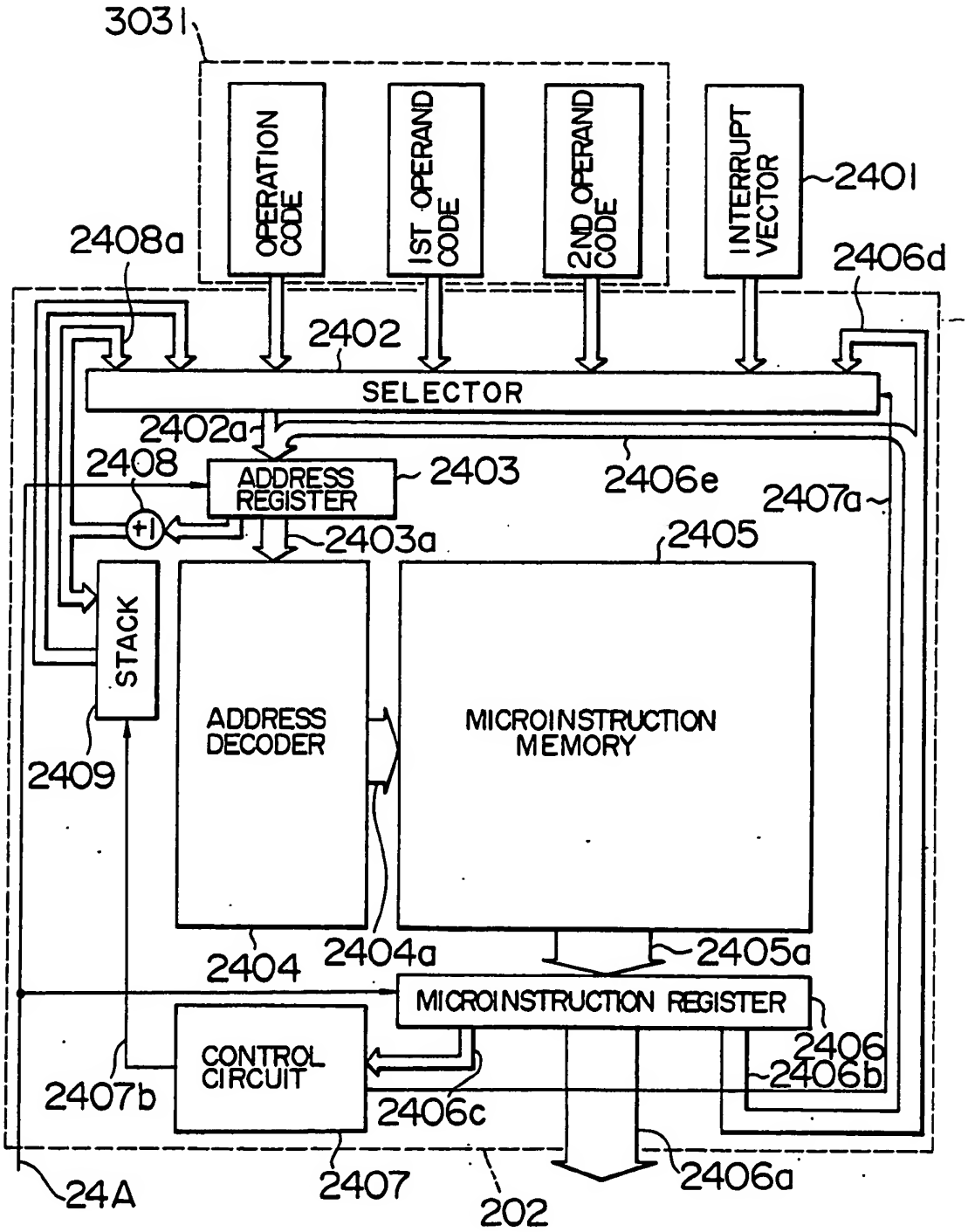

3031
OPERATION CODE
1ST OPERAND CODE
2ND OPERAND CODE
INTERRUPT VECTOR
2401
2408a
2406d
2402
SELECTOR
2402a
2408
ADDRESS REGISTER
2403
2406e
2407a
+1
2403a
2405
STACK
ADDRESS DECODER
MICROINSTRUCTION MEMORY
2409
2404
2404a
2405a
MICROINSTRUCTION REGISTER
2406
2406b
CONTROL CIRCUIT
2407b
2406c
24A
2407
202
2406a

FIG. 35

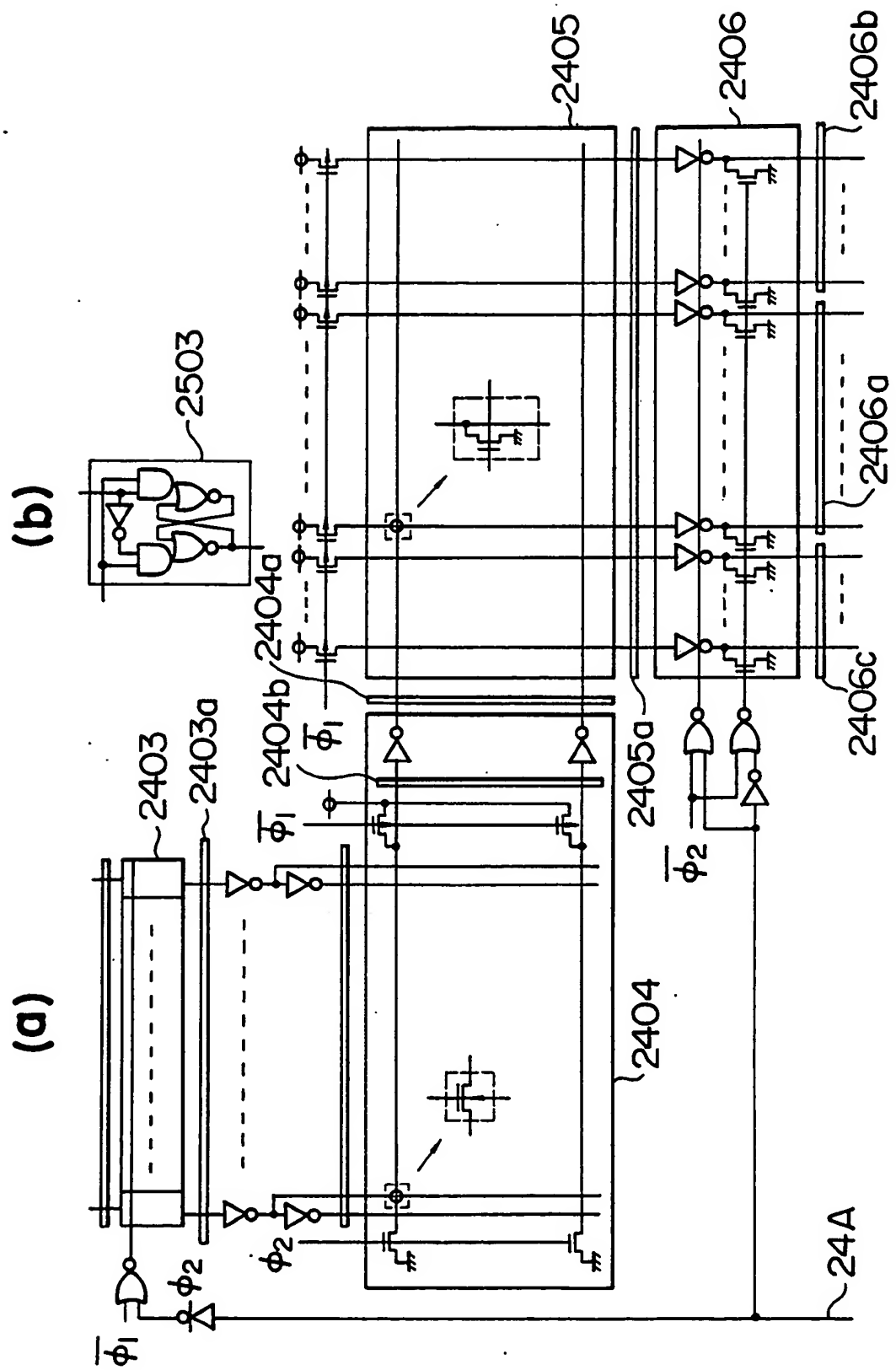

(a)
(b)
2503
2405
2406
2406b
2406a
2406c
2404a
2404b
2403
2403a
2405a
2404
24A
$\overline{\phi_1}$
$\phi_2$

FIG. 36

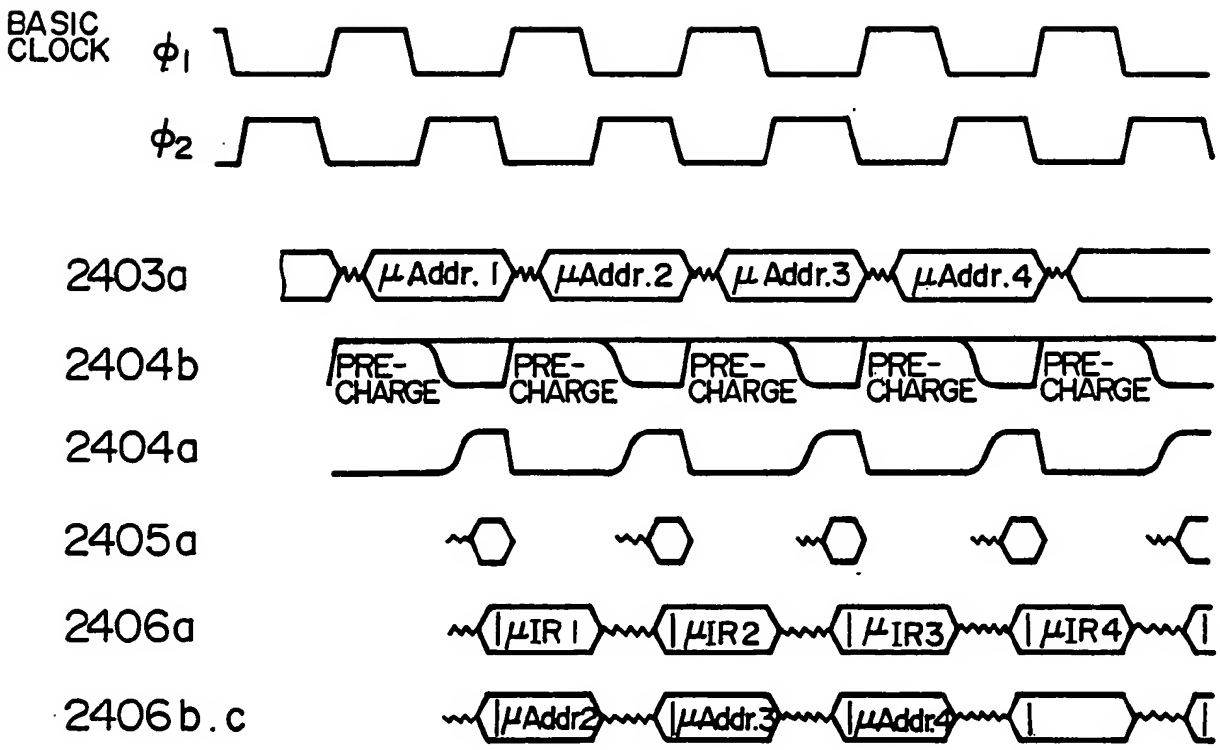

BASIC CLOCK
$\phi_1$
$\phi_2$
2403a
μAddr. 1
μAddr.2
μAddr.3
μAddr.4
2404b
PRE-CHARGE
PRE-CHARGE
PRE-CHARGE
PRE-CHARGE
PRE-CHARGE
2404a
2405a
2406a
μIR1
μIR2
μIR3
μIR4
2406b.c
μAddr2
μAddr.3
μAddr4

FIG. 37

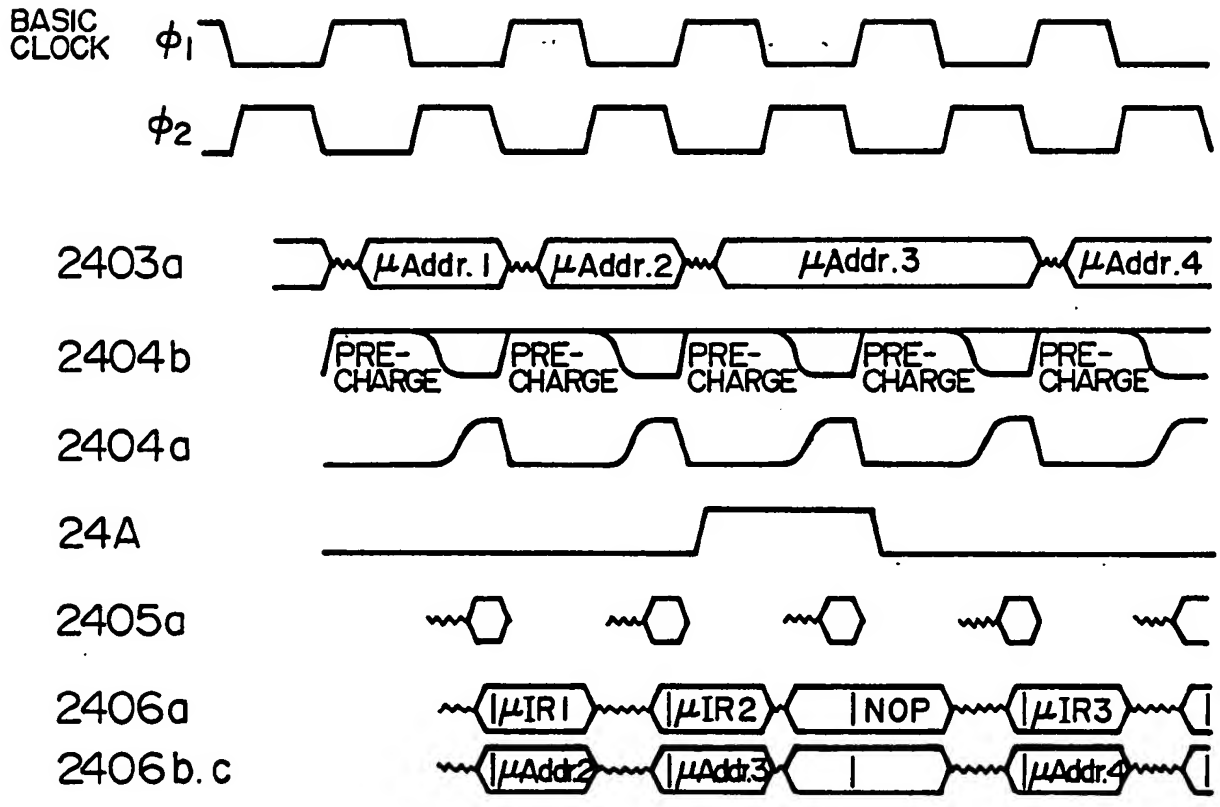

BASIC CLOCK
$\phi_1$
$\phi_2$
2403a
μAddr. 1
μAddr.2
μAddr.3
μAddr.4
2404b
PRE-CHARGE
PRE-CHARGE
PRE-CHARGE
PRE-CHARGE
PRE-CHARGE
2404a
24A
2405a
2406a
μIR1
μIR2
NOP
μIR3
2406b. c
μAddr2
μAddr.3
μAddr.4

FIG. 38

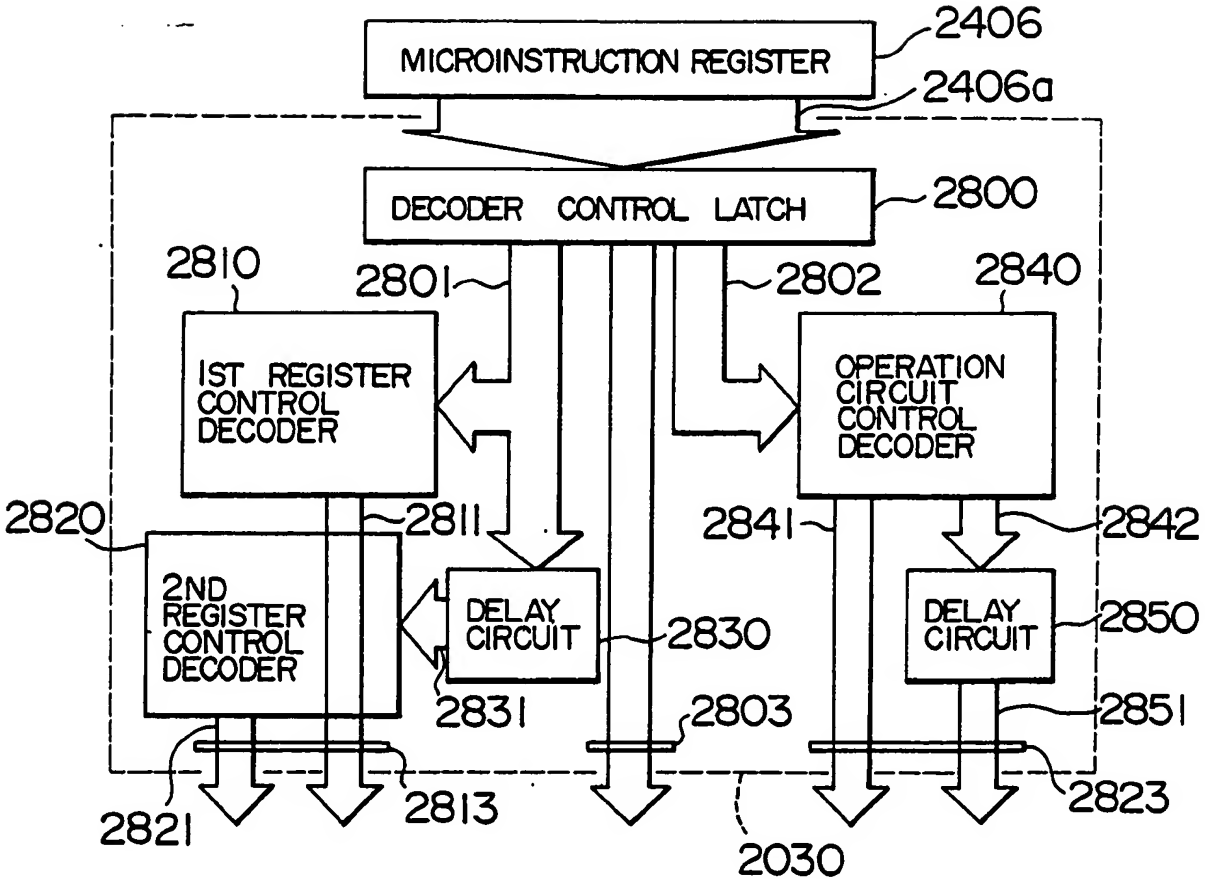

MICROINSTRUCTION REGISTER
2406
2406a
DECODER CONTROL LATCH
2800
2810
2801
2802
2840
IST REGISTER CONTROL DECODER
OPERATION CIRCUIT CONTROL DECODER
2820
2811
2841
2842
2ND REGISTER CONTROL DECODER
DELAY CIRCUIT
2830
DELAY CIRCUIT
2850
2831
2803
2851
2821
2813
2823
2030

FIG. 39

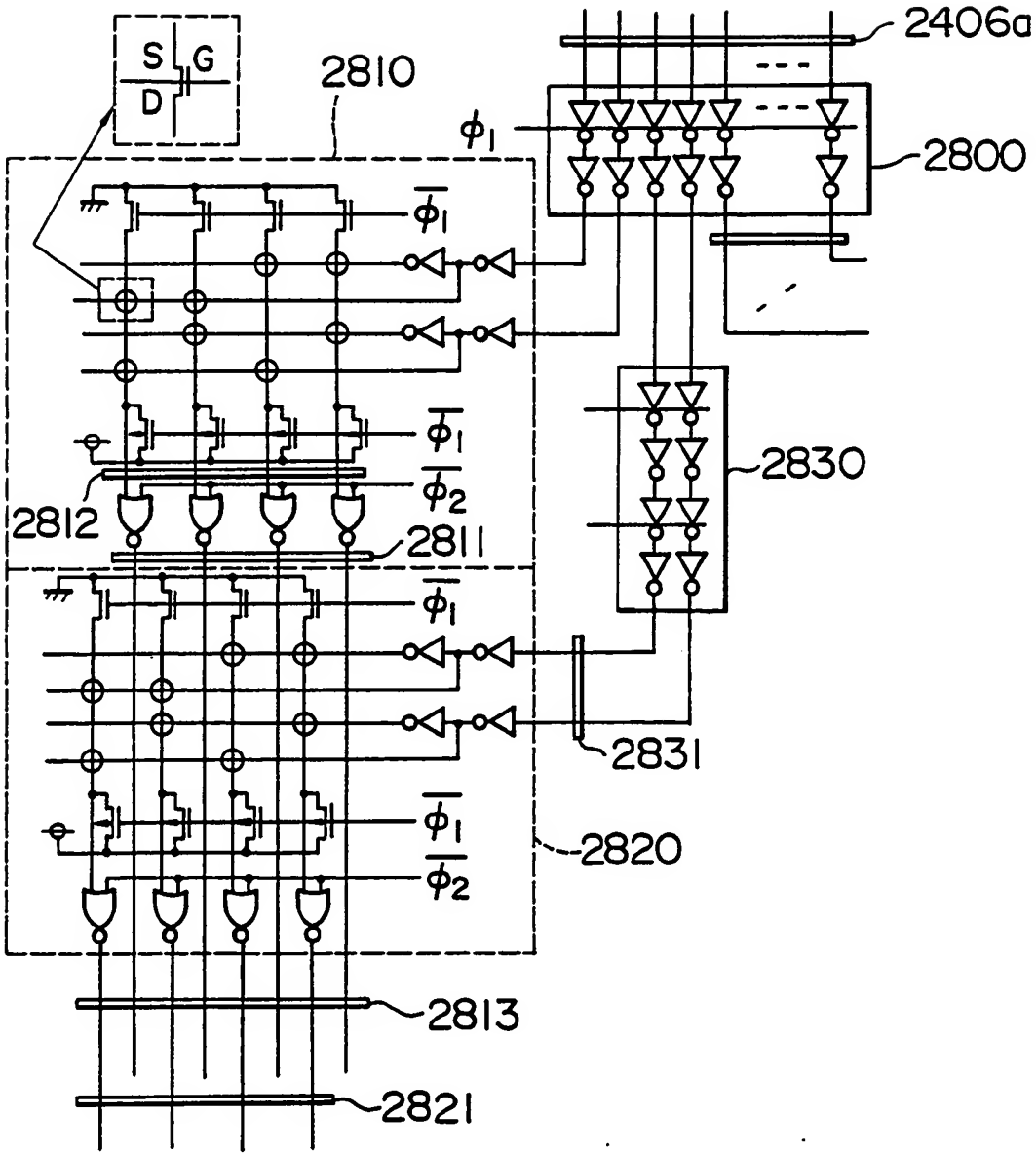

S
G
D
2810
2406a
$\phi_1$
2800
$\overline{\phi_1}$
$\overline{\phi_1}$
$\overline{\phi_2}$
2812
2811
2830
$\overline{\phi_1}$
2831
$\overline{\phi_1}$
$\overline{\phi_2}$
2820
2813
2821

FIG. 40

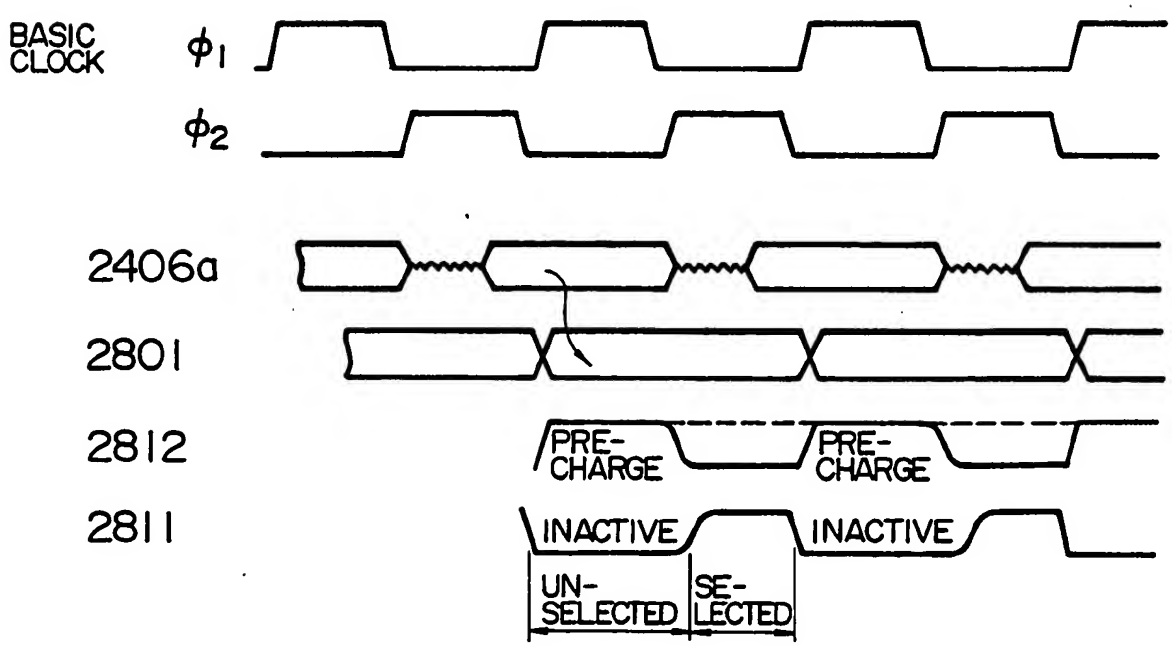

BASIC CLOCK
$\phi_1$
$\phi_2$
2406a
2801
2812
PRE-CHARGE
PRE-CHARGE
2811
INACTIVE
INACTIVE
UN-SELECTED
SE-LECTED

FIG. 41

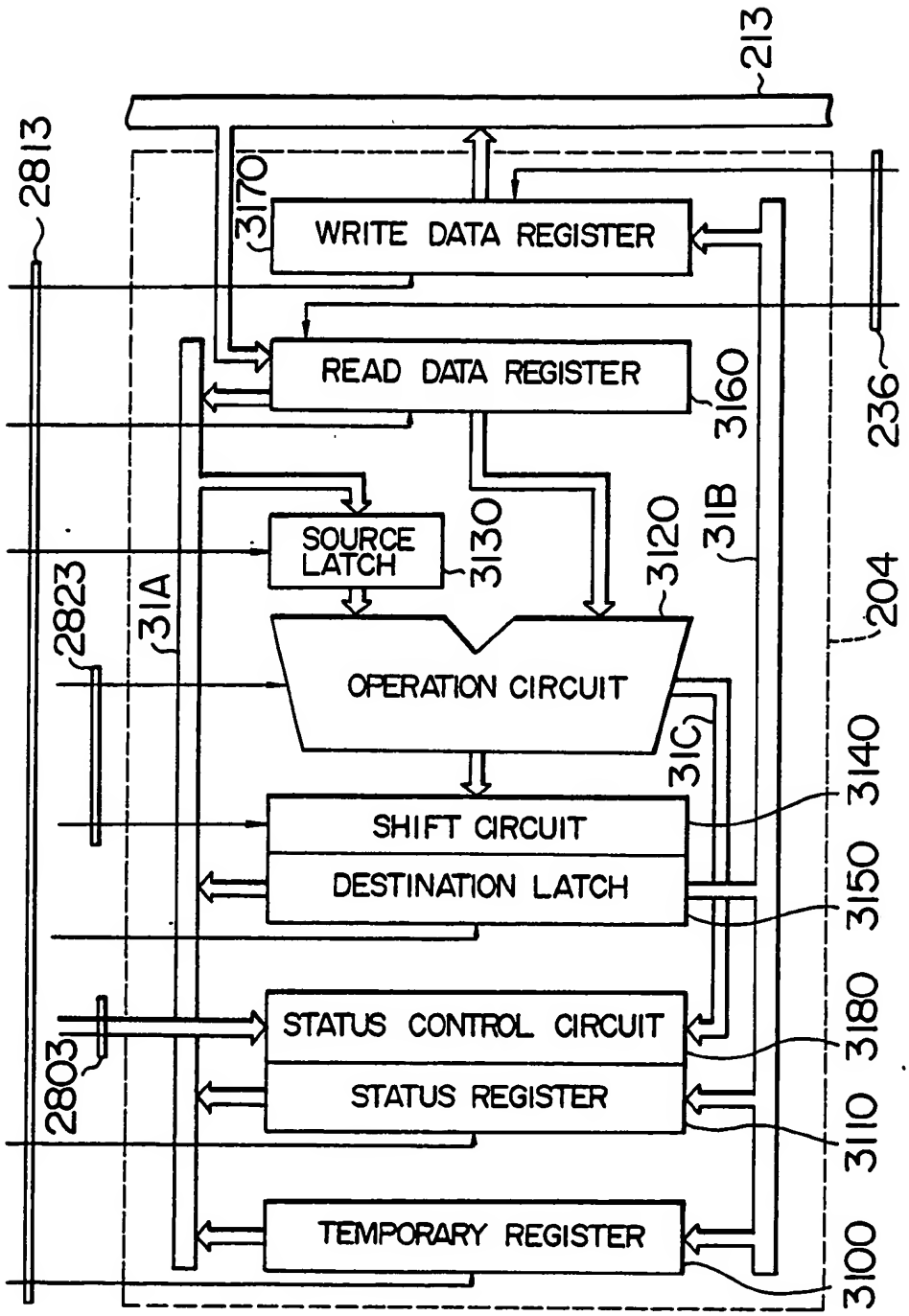

213
2813
3170
WRITE DATA REGISTER
READ DATA REGISTER
3160
236
SOURCE LATCH
3130
3120
31B
2823
31A
204
OPERATION CIRCUIT
31C
3140
SHIFT CIRCUIT
DESTINATION LATCH
3150
2803
STATUS CONTROL CIRCUIT
3180
STATUS REGISTER
3110
TEMPORARY REGISTER
3100

FIG. 42

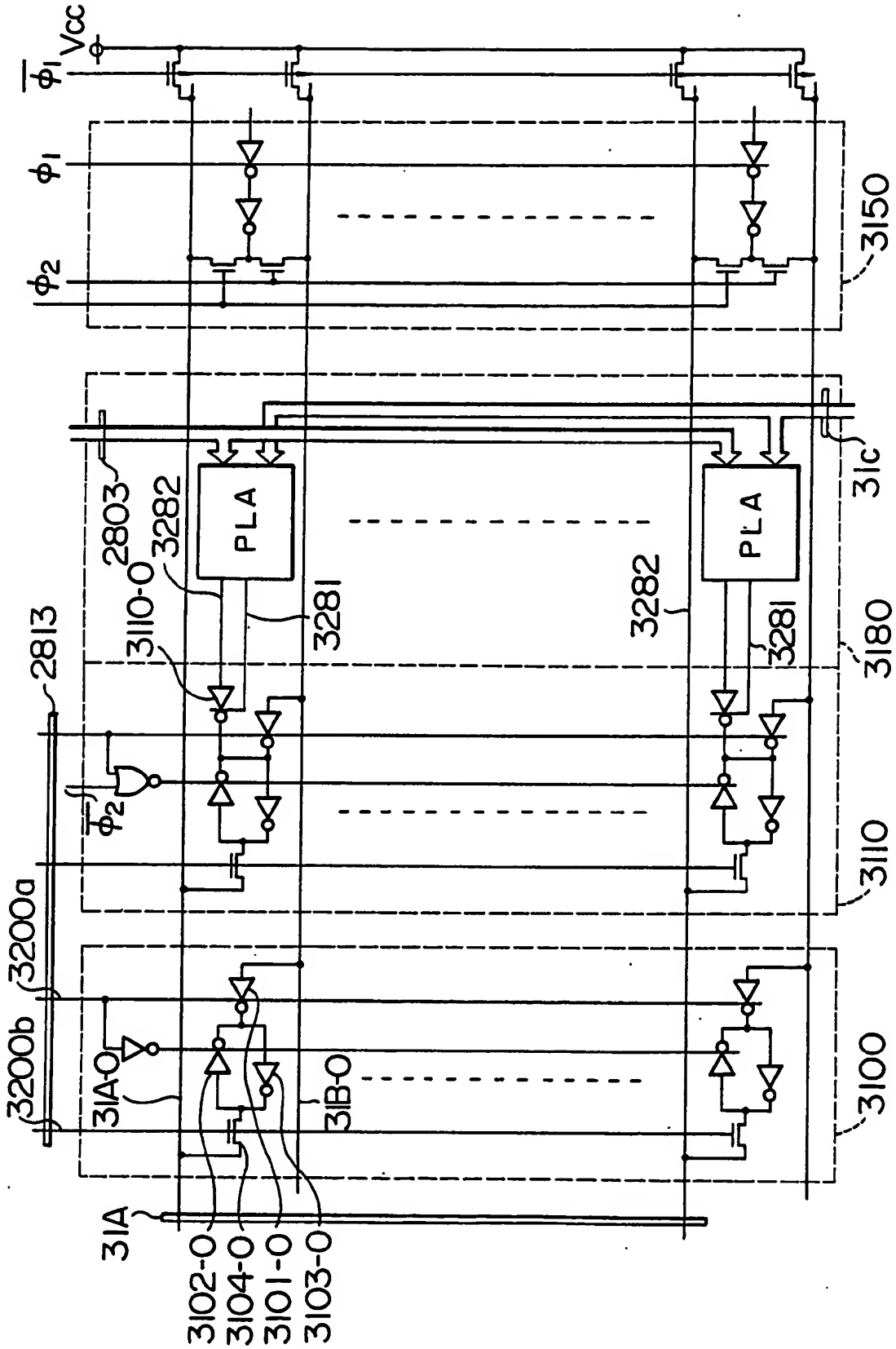

Vcc
$\overline{\phi_1}$
$\phi_1$
$\phi_2$
3150
2803
3282
PLA
3110-0
3281
2813
3180
3110
3200a
3200b
31A-0
31B-0
3100
31A
3102-0
3104-0
3101-0
3103-0
31c

FIG. 43

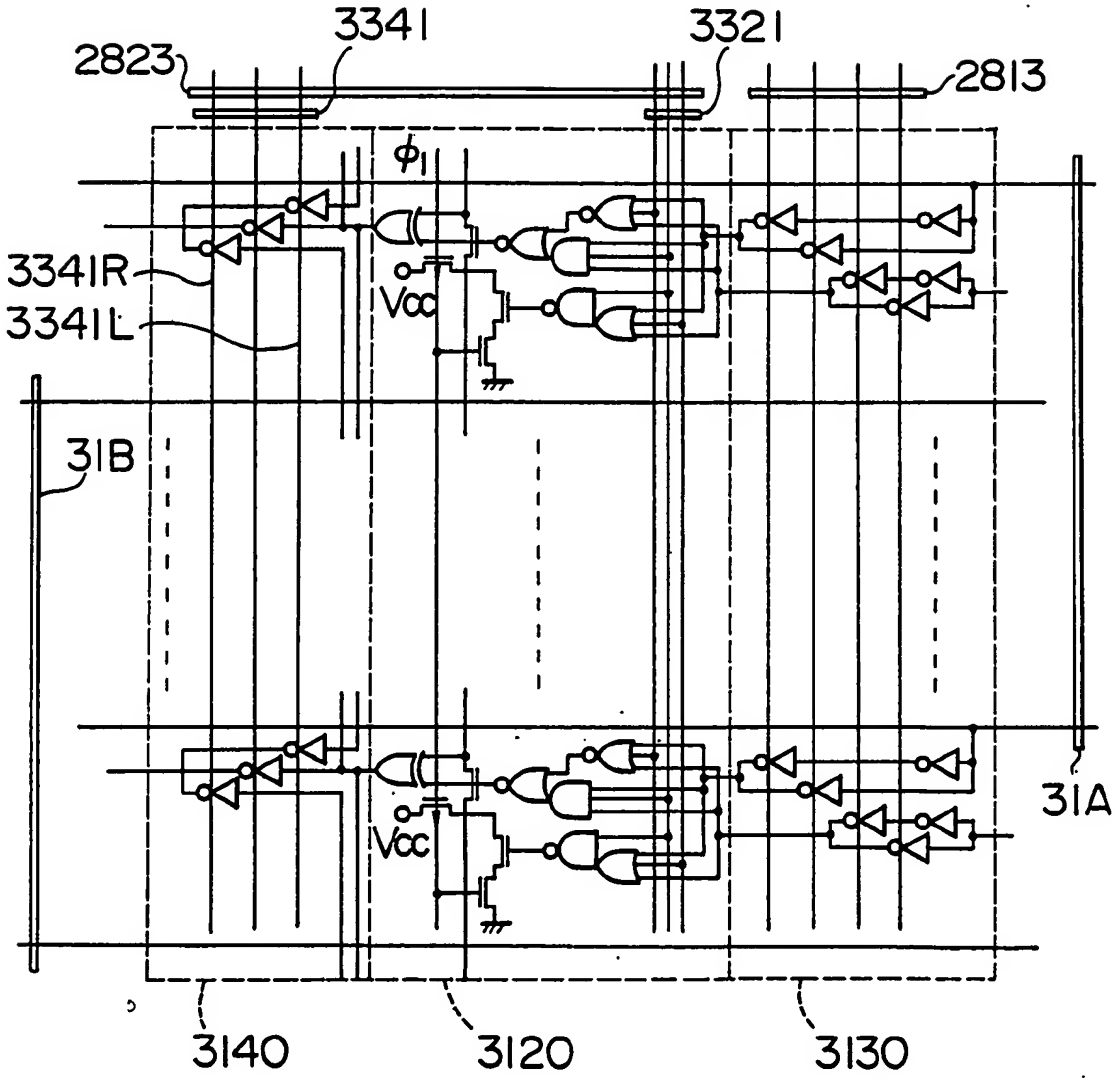

3341
3321
2823
2813
$\phi_1$
3341R
3341L
Vcc
31B
31A
Vcc
3140
3120
3130

FIG. 44

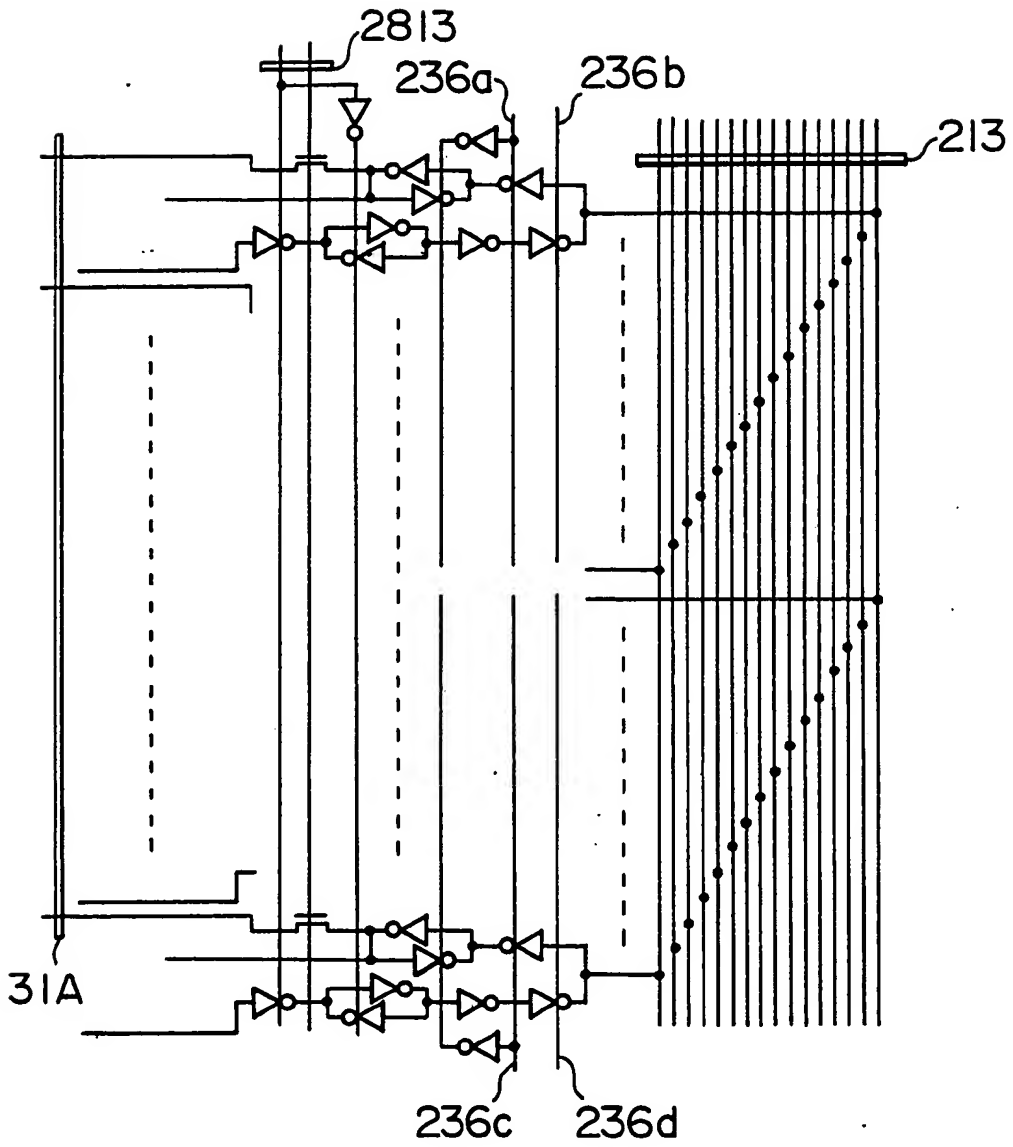


FIG. 45

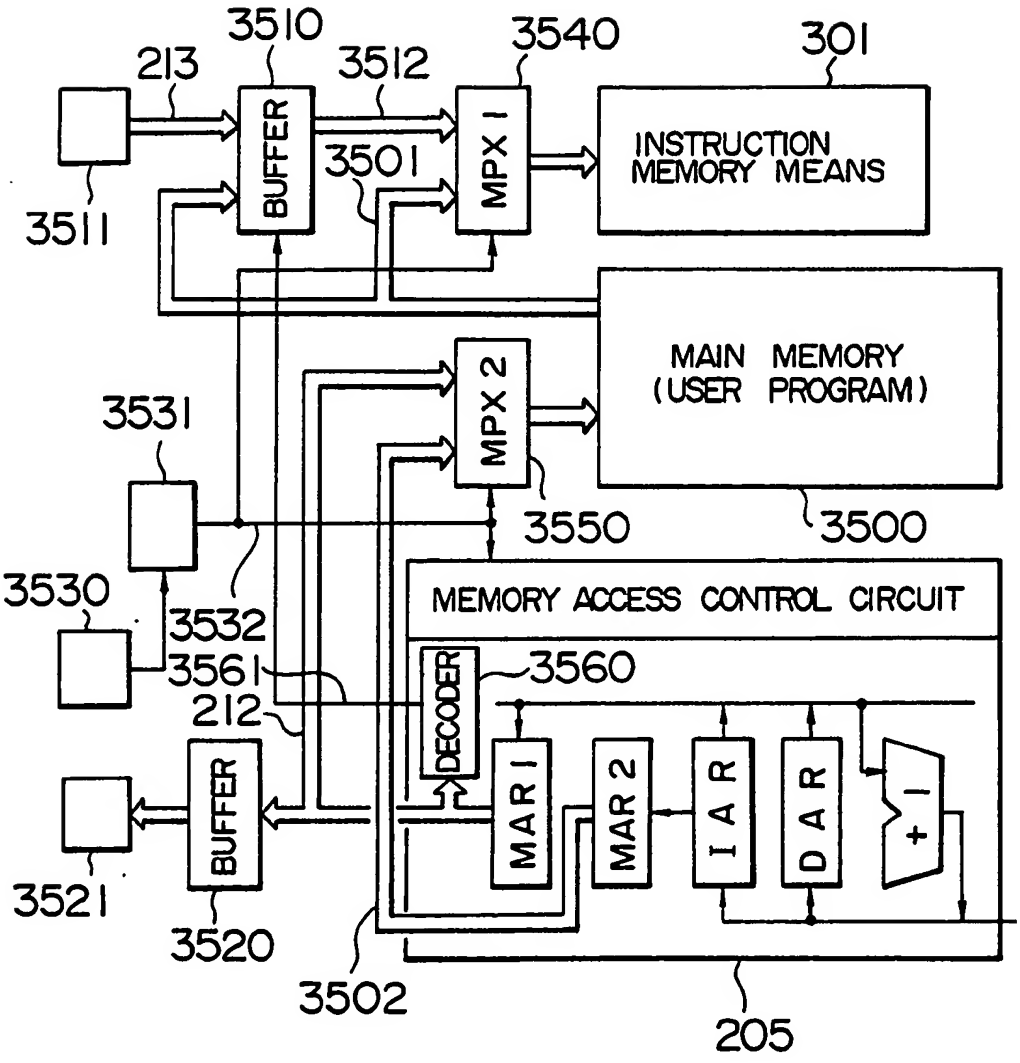

3510
213
3512
3540
301
BUFFER
MPX 1
INSTRUCTION MEMORY MEANS
3501
3511
MAIN MEMORY (USER PROGRAM)
MPX 2
3531
3550
3500
3530
MEMORY ACCESS CONTROL CIRCUIT
3532
3561
3560
DECODER
212
MAR 1
MAR 2
I A R
D A R
+ 1
BUFFER
3521
3520
3502
205

FIG. 46

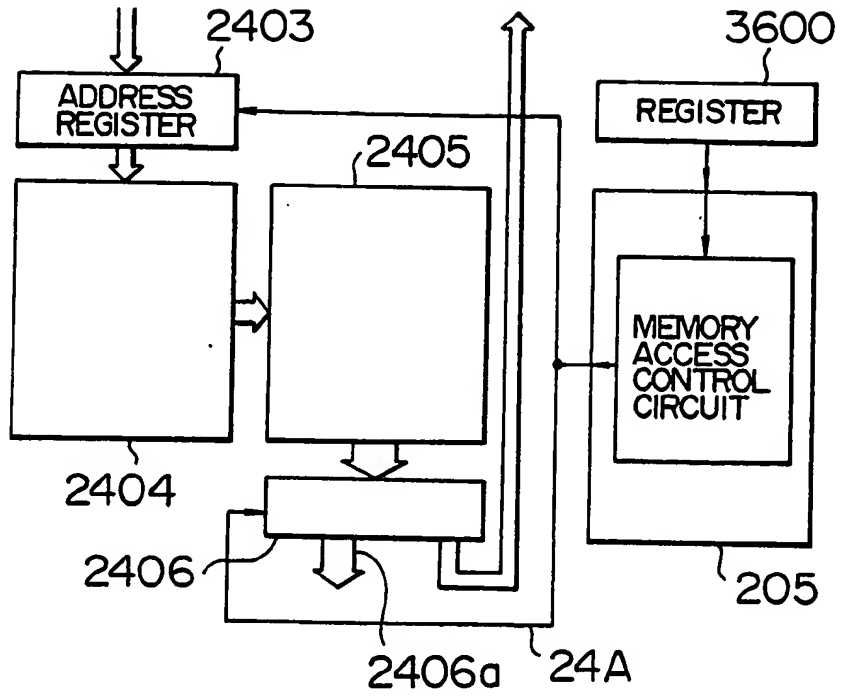
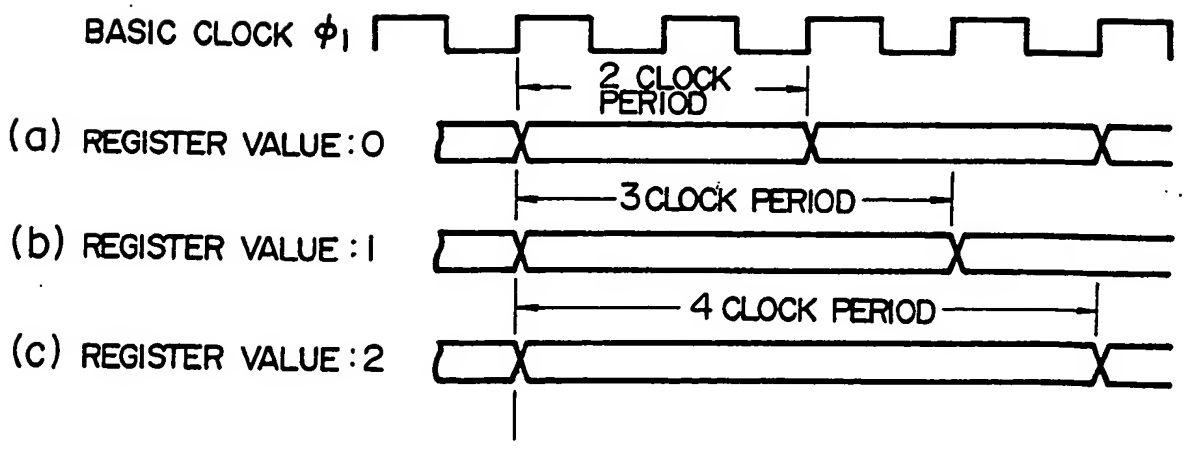

2403
3600
ADDRESS REGISTER
REGISTER
2405
MEMORY ACCESS CONTROL CIRCUIT
2404
2406
205
2406a
24A

FIG. 47


BASIC CLOCK φ1
2 CLOCK PERIOD
(a) REGISTER VALUE : 0
3 CLOCK PERIOD
(b) REGISTER VALUE : 1
4 CLOCK PERIOD
(c) REGISTER VALUE : 2

(19) Europäisches Patentamt
European Patent Office
Office européen des brevets

(11) Publication number: **0 199 173**
**A3**

(12) # EUROPEAN PATENT APPLICATION

(21) Application number: **86104747.0**

(22) Date of filing: **08.04.86**

(51) Int. Cl.⁴: **G06F 9/44 , G06F 9/42 , G06F 9/46**

(30) Priority: **08.04.85 JP 72646/85**
**14.08.85 JP 177542/85**

(43) Date of publication of application:
**29.10.86 Bulletin 86/44**

(84) Designated Contracting States:
**DE FR GB IT**

(88) Date of deferred publication of the search report:
**25.10.89 Bulletin 89/43**

(71) Applicant: **HITACHI, LTD.**
**6, Kanda Surugadai 4-chome Chiyoda-ku**
**Tokyo(JP)**

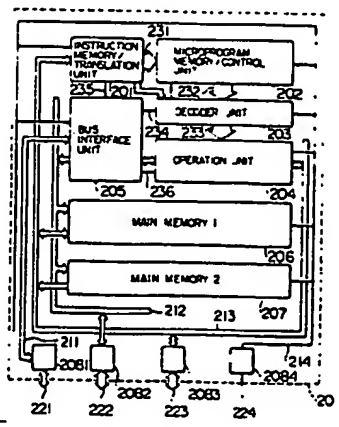
(72) Inventor: **Kida, Hiroyuki**
**17-1-102, Moriyamacho-3-chome**
**Hitachi-shi(JP)**
Inventor: **Maejima, Hideo**
**26-2, Nakanarusawacho-2-chome**
**Hitachi-shi(JP)**
Inventor: **Masuda, Ikuro**
**15-7, Moriyamacho-2-chome**
**Hitachi-shi(JP)**
Inventor: **Baba, Shirou**
**39-2-407, 5043, Yamaguchi**
**Tokorozawa-shi(JP)**

(74) Representative: **Patentanwälte Beetz sen. - Beetz jun. Timpe - Siegfried - Schmitt-Fumian**
**Steinsdorfstrasse 10**
**D-8000 München 22(DE)**

(54) **Data processing system.**

(57) A data processing system incorporating a main memory (206) for storing instructions and operands and performing data processing in a mode of microprogram control system in response to instructions read out of the main memory (208). The system translates an instruction word read out of the main memory (1, 2) into an intermediate machine word having the orthogonal format, and addresses a microprogram memory (202) in correspondence to the instruction word by analyzing the intermediate machine word. The system further incorporates a plurality of register sets so that each different task can use an individual register set, and a memory (2106) for memorizing the number of registers holding parameters used commonly among procedures corresponding to the register sets, so that the number of registers for each use can be changed arbitrarily for each register set by using the memory.

FIG. 1



European Patent Office

# EUROPEAN SEARCH REPORT

Application number

EP 86 10 4747

## DOCUMENTS CONSIDERED TO BE RELEVANT

| Category | Citation of document with indication, where appropriate, of relevant passages | Relevant to claim | CLASSIFICATION OF THE APPLICATION (Int Cl 4) |
|---|---|---|---|
| Y | WO-A-84 01 635 (IBM)<br>* Page 4, lines 15-25; page 5, lines 3-9; page 9, line 31 - page 10, line 17; figures 1-4 *<br>-- | 1-3,5, 8,9 | G 06 F 9/44<br>G 06 F 9/42<br>G 06 F 9/46 |
| Y | IBM TECHNICAL DISCLOSURE BULLETIN, vol. 15, no. 3, August 1972, page 920; New York, US<br>J.C. KEMP: "Instruction translator"<br>* Whole article *<br>-- | 1-3,5, 8,9 | |
| Y | ELECTRONIC DESIGN, vol. 32, no. 16, August 9, 1984, pages 229-234,236, 238,240; Waseca, MN, US<br>M. McBRIDE: "Microprogrammable chip set emulates mainframe processing"<br>* Page 229, right-hand column, line 25 - page 230, left-hand column, line 11; page 232, right-hand column, line 10 - page 233, left-hand column, line 10 *<br>-- | 1-3,5 | TECHNICAL FIELDS SEARCHED (Int. Cl.4)<br>G 06 F |
| Y | EP-A-0 087 008 (IBM)<br>* Page 1, lines 18-28; figure 1 *<br>-- | 3 | |
| Y | GB-A-2 002 553 (ITEK)<br>* Figure 6; page 3, lines 61-70; page 4, lines 7-32 *<br>-- | 5 | |
| A | PATENT ABSTRACTS OF JAPAN, vol. 7, no. 73 (P-186)(1218), March 25, 1983 & JP-A-58 03 040 (NIPPON DENKI K.K.) 08-01-1983<br>./. | | |

The present search report has been drawn up for all claims

| Place of search | Date of completion of the search | Examiner |
|---|---|---|
| THE HAGUE | 05-07-1989 | WILTINK |

CATEGORY OF CITED DOCUMENTS

X : particularly relevant if taken alone
Y : particularly relevant if combined with another document of the same category
A : technological background
O : non-written disclosure
P : intermediate document

T : theory or principle underlying the invention
E : earlier patent document, but published on, or after the filing date
D : document cited in the application
L : document cited for other reasons

& : member of the same patent family, corresponding document

EPO Form 1503 03 82

European Patent Office

## CLAIMS INCURRING FEES

The present European patent application comprised at the time of filing more than ten claims.

☐ All claims fees have been paid within the prescribed time limit. The present European search report has been drawn up for all claims.

☐ Only part of the claims fees have been paid within the prescribed time limit. The present European search report has been drawn up for the first ten claims and for those claims for which claims fees have been paid.

namely claims:

☐ No claims fees have been paid within the prescribed time limit. The present European search report has been drawn up for the first ten claims.

## X LACK OF UNITY OF INVENTION

The Search Division considers that the present European patent application does not comply with the requirement of unity of invention and relates to several inventions or groups of inventions.

namely:

1. Claims 1-10: Translate instructions to intermediate form, decode with microprogram
2. Claims 11,12: Instruction register with half-width readout
3. Claims 13-19: Multiple register sets with overlap
4. Claims 20-23: Halt microprogram by inhibiting renewal of instruction- or address-register
5. Claims 24,25: Decode instructions by means of PLA's

☒ All further search fees have been paid within the fixed time limit. The present European search report has been drawn up for all claims.

☐ Only part of the further search fees have been paid within the fixed time limit. The present European search report has been drawn up for those parts of the European patent application which relate to the inventions in respect of which search fees have been paid.

namely claims:

☐ None of the further search fees has been paid within the fixed time limit. The present European search report has been drawn up for those parts of the European patent application which relate to the invention first mentioned in the claims.

namely claims:

European Patent Office

# EUROPEAN SEARCH REPORT

Application number

EP 86 10 4747



## DOCUMENTS CONSIDERED TO BE RELEVANT

| Category | Citation of document with indication, where appropriate, of relevant passages | Relevant to claim | CLASSIFICATION OF THE APPLICATION (Int Cl 4) |
|---|---|---|---|
| | * Abstract * | 1-3,5 | |
| | -- | | |
| X | IBM TECHNICAL DISCLOSURE BULLETIN, vol. 24, no. 3, August 1981, pages 1401-1403; New York, US<br>F.T. BLOUNT et al.: "Shared instruction buffer for multiple instruction streams" | | |
| | * Page 1401, lines 6-10; figure 1 * | 11 | |
| Y | | 12 | |
| | -- | | |
| Y | PATENT ABSTRACTS OF JAPAN, vol. 8, no. 215 (P-305)(1652), October 2, 1984<br>& JP-A-59 99 552 (MATSUSHITA DENKI SANGYO K.K.) 08-06-1984 | | |
| | * Abstract * | 12 | |
| | -- | | TECHNICAL FIELDS SEARCHED (Int. Cl 4) |
| A | IBM TECHNICAL DISCLOSURE BULLETIN, vol. 19, no. 12, May 1977, page 4645, New York, US<br>D. BAZLEN et al.: "Instruction buffer loading " | | |
| | * Whole article * | 11,12 | |
| | -- | | |
| A | IBM TECHNICAL DISCLOSURE BULLETIN, vol. 24, no. 7B, December 1981, pages 3896-3897; New York, US<br>A. BLUM: "Multiple general purpose register sets embedded in processor chips" | | |
| | * Figure 2, page 3897, lines 16-29 * | 13,15, 16 | |
| | -- ./.. | | |

The present search report has been drawn up for all claims

| Place of search | Date of completion of the search | Examiner |
|---|---|---|
| | | |

CATEGORY OF CITED DOCUMENTS

X : particularly relevant if taken alone
Y : particularly relevant if combined with another document of the same category
A : technological background
O : non-written disclosure
P : intermediate document

T : theory or principle underlying the invention
E : earlier patent document, but published on, or after the filing date
D : document cited in the application
L : document cited for other reasons

& : member of the same patent family, corresponding document

EPO Form 1503 03 82

European Patent Office

# EUROPEAN SEARCH REPORT

Application number

EP 86 10 4747



## DOCUMENTS CONSIDERED TO BE RELEVANT

| Category | Citation of document with indication, where appropriate, of relevant passages | Relevant to claim | CLASSIFICATION OF THE APPLICATION (Int Cl 4) |
|---|---|---|---|
| A | GB-A-1 233 270 (IBM)<br>* Page 1, lines 32-46 * | 13 | |
| A | US-A-4 410 939 (KAWAKAMI)<br>* Figure 8; column 9, line 35 - column 10, line 19 * | 13,14 | |
| A | EP-A-0 061 324 (ZILOG)<br>* Figure 1; page 4, line 4 - page 5, line 20 * | 13,14, 17 | |
| A | US-A-4 352 157 (NAMIMOTO)<br>* Column 8, line 29 - column 9, line 23; column 12, line 35 - column 13, line 27; figures 7,12 * | 13,18, 19 | |
| A | COMPUTER, September 1982, pages 8-18,20,21; IEEE, New York, US D.A. PATTERSON et al.: "A VLSI risc"<br>* Figure 3; page 12, left-hand column, line 3 - right-hand column, line 19 * | 13,18, 19 | TECHNICAL FIELDS SEARCHED (Int. Cl 4) |
| Y | US-A-3 736 567 (LOTAN et al.)<br>* Figure 1; column 4, lines 27-68 * | 20-23 | |
| Y | EP-A-0 025 855 (IBM)<br>* Figures 3,4; page 5, line 9 - page 6, line 5 * | 20-23 | |

./...

The present search report has been drawn up for all claims

| Place of search | Date of completion of the search | Examiner |
|---|---|---|
| | | |

CATEGORY OF CITED DOCUMENTS

- X : particularly relevant if taken alone
- Y : particularly relevant if combined with another document of the same category
- A : technological background
- O : non-written disclosure
- P : intermediate document
- T : theory or principle underlying the invention
- E : earlier patent document, but published on, or after the filing date
- D : document cited in the application
- L : document cited for other reasons
- & : member of the same patent family, corresponding document

EPO Form 1503 03 82

European Patent Office

# EUROPEAN SEARCH REPORT

Application number

EP 86 10 4747



## DOCUMENTS CONSIDERED TO BE RELEVANT

| Category | Citation of document with indication, where appropriate, of relevant passages | Relevant to claim | CLASSIFICATION OF THE APPLICATION (Int Cl 4) |
|---|---|---|---|
| A | PATENT ABSTRACTS OF JAPAN, vol. 6, no. 245 (P-159)(1123), December 3, 1982<br>& JP-A-57 141 757 (NIPPON DENKI K. K.) 02-09-1982<br>* Abstract * | 20-23 | |
| Y | EP-A-0 031 889 (IBM)<br>* Figure 1; page 10, line 37 - page 11, line 16 * | 24,25 | |
| Y | EP-A-0 087 601 (IBM)<br>* Figures 6A,6B; page 21, lines 3-10,22-27; page 24, lines 6-15 * | 24,25 | |
| Y | US-A-3 974 366 (HEBENSTREIT)<br>* Figure 3; column 4, lines 7-16 * | 24,25 | TECHNICAL FIELDS SEARCHED (Int Cl 4) |
| A | MICROPROCESSING AND MICROPROGRAMMING, vol. 12, no. 1, August 1983, pages 15-31; North-Holland Publ. Co., Amsterdam, NL<br>W. GRASS: "A synthesis system for PLA-based programmable hardware"<br>* Figure 3 * | 25 | |

The present search report has been drawn up for all claims

| Place of search | Date of completion of the search | Examiner |
|---|---|---|
| | | |

CATEGORY OF CITED DOCUMENTS

X : particularly relevant if taken alone
Y : particularly relevant if combined with another document of the same category
A : technological background
O : non-written disclosure
P : intermediate document

T : theory or principle underlying the invention
E : earlier patent document, but published on, or after the filing date
D : document cited in the application
L : document cited for other reasons

& : member of the same patent family, corresponding document

EPO Form 1503 03 82